COMPUTER

IX-X

PUNJAB TEXTBOOK BOARD, LAHORE

APPEAL

Punjab Textbook Board is your own organization. It produces quality and cost effective textbooks for the students. These textbooks are produced under the supervision of experienced educationists. The textbooks are developed in such a manner that they enhance creative faculties in the children.

Unfortunately, some of the private publishers produce Guide-books, Class-notes, Testpapers, and similar kind of books which contain highly sub-standard textual material. There is no doubt that children cram this material and get-through their school examination. But this is a fact that this kind of material does not nourish the creative capabilities in the students that is why these children simply stuck-up in the professional examinations at higher level.

Respected parents, teachers and dear students, you are not bound to buy un-authorized books i.e, test papers, guide-books, class-notes etc.

If somebody compels you to buy any of the above books, please do write to the **Chairman Punjab Textbook Board** on the address given below.

For Classes-IX-X, the following textbooks are approved by the Ministry of Education, (Curriculum Wing) Government of Pakistan and prescribed by Education Department, Govt. of Punjab.

1. English Book - I
2. English Book - II
3. New English Gramar & Composition
4. Muraqa-e-Urdu
5. Urdu Qawaid-o-Insha
6. Pakistan Studies
7. Islamiyat
8. Geography for foreign students
9. Islamiyat
10. Al Mutalia-tul-Arabia
11. Civics(Shehriat)
12. Farsi
13. Punjabi Adab
14. Tareekh-e-Pakistan
15. Geographia
16. Mubadiat-e-Taleem
17. Work Book Geographia
18. Urdu
19. Chemistry
20. Physics
21. Biology
22. Mathematics
23. Mubadiat-e-Home Economics
24. General Science
25. Art
26. Computer Studies
27. Bakery & Confectionery
28. Nursing & First Aid
29. English Type Writing
30. Hisabat Khana Dari,Wa Mutalqa Masail
31. Libas Sazi
32. Haath,Wa Machine ki Bunai aur Kroshia
33. Geometrical and Technical Drawing
34. Khilonay aur Guria Sazi
35. Business Method
36. Furniture and Cabinet Making
37. Wood Carving in Lay Work
38. Refrigeration and Air Conditioning
39. Zari Paidawar Work Book
40. Punjabi Adab Work Book
41. Secretarial Practice

These textbooks are published and marketed by Punjab Textbook Board, carrying its seal given here.

Muhammad Athar Tahir (Tamgha-i-Imtiaz)
CHAIRMAN
Punjab Textbook Board,
21-E-II, Gulberg III,
Lahore.

A TEXTBOOK OF COMPUTER STUDIES

For

Classes IX-X

PUNJAB TEXTBOOK BOARD LAHORE

"Copy Rights Reserved with the N.W.F.P. Textbook Board, Peshawar."

Produced by:

N.W.F.P. Textbook Board, Peshawar.

Approved by:

Federal Ministry of Education, Government of Pakistan, Islamabad as sole textbook of schools throughout Pakistan.

Authors:

Dr. Muhammad Abid (Convener) University of Peshawar.
Mr. Attaullah.
Mr. Saeed Mehfooz.
Mr. Abdul Rahim.
Mr. Abdul Hameed Toor
Mr. Muhammad Afzal
M. Tariq Siddiqui,

Editor:

Mazhar Hayat

Supervision : **Mazhar Hayat**

Editor - in - Chief : Mrs. Nazma Usman

Director Technical : **Prof. Naseer-ud-Din.**

Publisher : Daizy Packages Lahore

Printer : M.A.S. Printers, Karim Park Lahore.

This book is prepared under joint venture of all Pakistan Text Book Boards. The Electronic Composing and Processing of this book is done by Izharsons for Punjab Text Book Board, Lahore.

PREFACE

In this century, due to tremendous research in various scientific fields, there have been some marvellous inventions. In this race of research and invention, computer is also effectively contributing its share. As a matter of fact it can be asserted that in the recent past most of the scientific inventions and discoveries were dependent on the computers.

Due to advancement of technology and by the use of computer, it has been possible for a man to launch satellites and land on the surface of the moon. The methodology of teaching the subject has also become of paramount importance. The Govt. of Pakistan and persons at the helm of affairs are also seized this efficient problem and the work of curriculum development in this subject is going on for the last couple of years.

The authors have worked very hard to develop this book, but due to extremely short time at their disposal and circumstances beyond this control, it cannot be claimed that the book is totally free from errors and omission.

We are introducing this book for the secondary stage. The teachers, students and other readers are requested to go through this book minutely because their remarks and views regarding this book would improve our future publication because there is always a room for improvement.

Editor

TABLE OF CONTENTS

CHAPTER-1	History and Development of Computer	1-7
CHAPTER-2	Microcomputer	8-12
CHAPTER-3	Input/Output and Backing Storage Devices	13-19
CHAPTER-4	Introduction to Computer Software	20-25
CHAPTER-5	Operating Systems	26-33
CHAPTER-6	Number Systems	34-40
CHAPTER-7	Problem-Solving and Flow Charting	41-46
CHAPTER-8	Programming in Basic Language	47-53
CHAPTER-9	Input and Output Statements	54-60
CHATPER-10	Transfer of Control Statement	61-67
CHAPTER-11	Repetitive Statements	68-72
CHAPTER-12	Arrays	73-77
CHAPTER-13	Subroutines and Functions	78-88
CHAPTER-14	Graphics	89-96

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

Chapter 1

HISTORY AND DEVELOPMENT OF COMPUTERS

1.0 Introduction

A computer is an electronic device that stores, retrieves and processes information. It helps us to solve wide and varied problems accurately and quickly. These problems may relate to the fields such as industry, commerce, engineering and education.

Computer is extremely an efficient and reliable machine. It has the advantage over human mind that it does not get bored of repetitions, and performs a function each time with the same speed and accuracy.

1.1 Historical Background

Throughout history, man has been looking constantly for ways of making calculations easier.

For this purpose Chinese made a wooden frame holding a number of parallel rods on which beads or stones were threaded. This device was named as "abacus" or "counting frame". An abacus is considered as the first recognized manual device used for counting and doing simple arithmetic calculations.

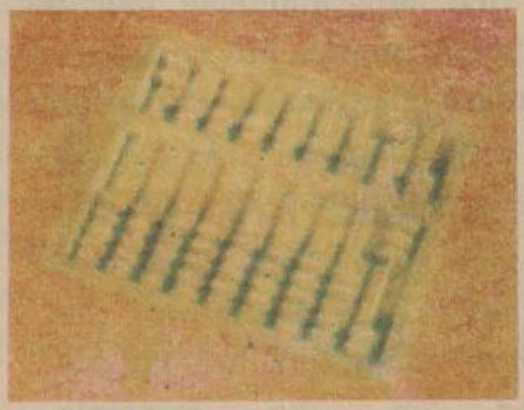


Fig 1.1 An Abacus

With the introduction of Arabic numerals from 0 to 9, the calculation process became much easier. Using these numbers, John Napier developed a calculating device. Which was named as Napier's Bones. Later on, this device led to the development of another calculating device called the slide rule. This slide rule in its improved shape was in use till the introduction of electronic calculators.

In 1642, Blaise Pascal invented the first mechanical calculating machine, comprising of wheels, dials and windows through which the numbers could be read. The results could be obtained upto eight digit numbers.

Fig 1.2 Pascal's Calculator

In 1671, Gottfried W. Von Leibnitz invented a better calculating machine. Pascal's machine could only add and subtract, but Leibnitz's machine could also multiply, divide and find square roots.

Although there were a number of improvements in calculating devices, no conceptual changes were recorded till the end of 18th century.

In 1801, Joseph Marie Jecquard, a French engineer, developed punched card system for the power-looms. This system helped in creating specific weaving patterns on cloths automatically. This system was later on used with computing devices.

In 1823, Charles Babbage, a British mathematician worked on the construction of a huge mechanical machine based on the principle of difference of the squares of numbers. He named the machine as difference engine. In 1833, while working on this machine, he envisioned and designed the first programmable machine and he called his planned machine the analytical engine, but he never converted his dreams into reality, because the standard of engineering & technology was not so high at that time.

In fact Babbage's machine gave birth to the research in the field of computers, and that is why, the modern computers are largely based on the principle developed by Babbage. It is why Babbage is considered as father of modern computer.

Fig 1.3 Part of Babbage's Analytical Engine

In 1890, an American, Herman Hollerith designed the

first electric computing device. He also made use of punched cards in this machine.

The efforts to improve the computing machine continued. In 1948, a success came in this field when Howard H. Aiken developed an electro-mechanical computer and named it Mark-1. Mark-1 is considered to be one of the first true computers because it could store informations and instructions.

Fig 1.4 MARK-1

In 1946, John W. Mauchly and J.P. Eckert developed an Electronic Numerical Integrator And Calculator (ENIAC). ENIAC was improved by John Von Neumann in 1950 and the improved version was named as EDVAC (Electronic Discrete Variable Automatic Computer). EDVAC was the first computer that could store both the program as well as the data. By 1950, computers with automatic data processing capabilities were developed. One such computer was UNIVAC-I.

1.2 Generations of Computers

It is quite obvious that the development of computer is a continuous process. However, the development of computers could be categorized into the following generations based on the technology used for the machine.

1.2.1 First Generation (1945-55)

The computers, ENIAC, EDVAC, UNIVAC and Mark-1, developed in the initial phase of computer development belong to the first generation of computers. These computers were mostly made between 1945 and 1955. Computers developed in this generation used the vacuum tubes. These computers were the fastest calculating devices of their time. However, the memory size of these computers was limited. They were bulky in size, costly and unreliable.

1.2.2 Second Generation (1955-64)

Transistors replaced vacuum tubes and gave birth to the second generation of computers like UNIVAC II, IBM 1401 and CDC 1604. These computers were smaller in size, faster and more reliable as compared to the first generation computers.

1.2.3 Third Generation (1964-70)

In early 1960, the electronic technology of solid-state was introduced. The development of Integrated Circuits (ICs)* is referred to as solid-state technology. The computers using this technology belong to the third generation. This generation of computers was even more reliable, faster and smaller than the previous ones.

1.2.4 Fourth Generation (1970 and Onwards)

In 1970s, the integrated circuit technology was developed sufficiently to integrate all main functions of a computer on a single chip called a microprocessor. The introduction of microprocessors brought the computer age into fourth generation. A direct result is the development of microcomputers which brought computers down from the organizational level to the personal level.

1.3 Types of Computers

Computers can be classified into three types. This classification is based on the hardware structure and the way through which physical quantities are represented in a computer. The three types are:

**The Integrated Circuits (ICs) are the collection of many electronic devices, like transistors on a single chip (small piece) of silicon.*

1. Analog Computer 2. Digital Computer
3. Hybrid Computer

These types of computers are explained below:

1.3.1 Analog Computers

An analog computer is the one that measures or processes data in a continuous form. A traditional watch or clock is an analog device in which time is represented in a continuous fashion.

The analog computers represent physical quantities like speed, weight, temperature etc. through an analogy with the electrical properties of current and voltage. Analog computers are the first computers being developed and provided the basis for the development of modern digital computers. These computers are mainly made of electrical devices like resisters, amplifiers and transistors. These are used for the calculation and observation of physical quantities or their measurement such as temperature and speed. In general, analog computers are very fast in processing and are used for real-time simulation.

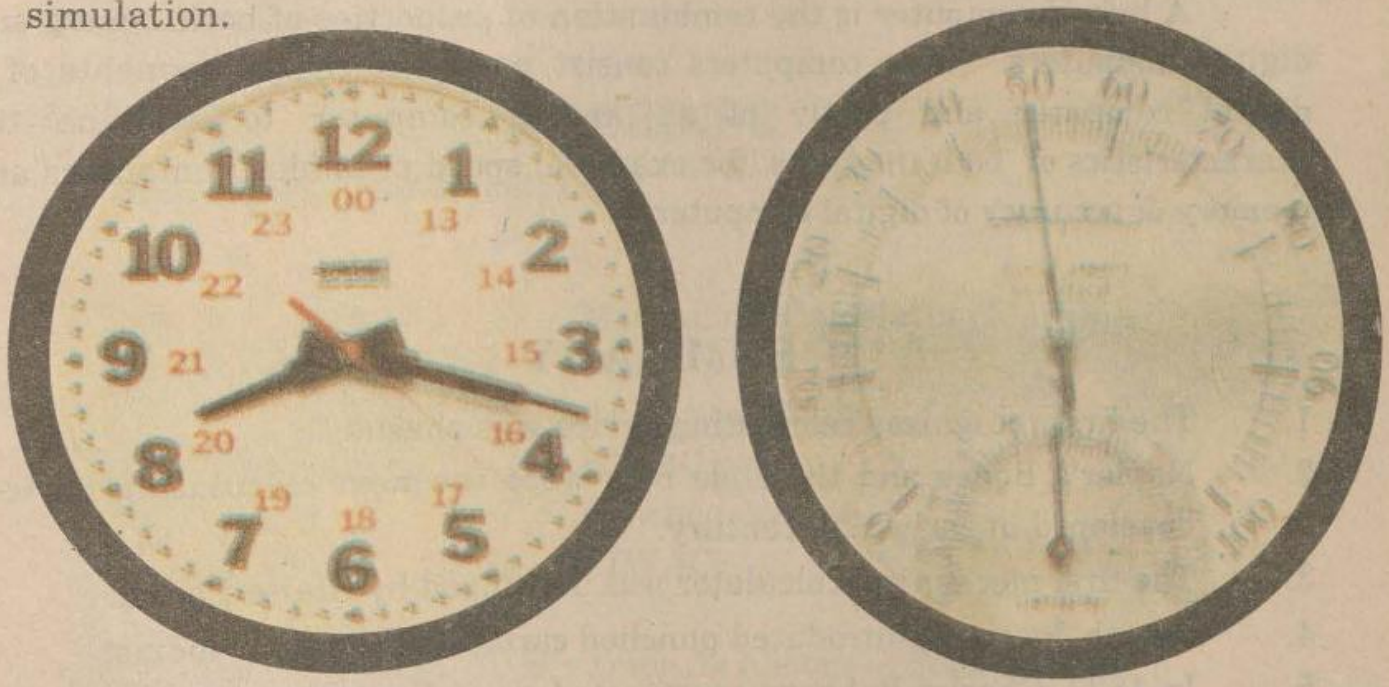


Fig. 1.2 (a) Analog Clock *(b) Analog Thermometer and Hydrometer*

1.3.2 Digital Computers

Digital computers represent physical quantities with the help of digits or numbers. These numbers are used to perform arithmetic calculations and also make logical decisions to reach a conclusion depending on the data they receive from the user. In general, digital computers are easy to program and

are used for general purposes. All the personal computers belong to this category.

Digital Computers can be classified into three groups:

1. Mainframe computer.
2. Minicomputer.
3. Microcomputer.

This classification is based on the processing power, functional capabilities and size of memory etc.

Microcomputer is the smallest in range having very small memory and processing capabilities. The major components of a microcomputer will be discussed in the next chapter.

Fig. 1.3 (A Digital Clock)

1.3.3 Hybrid Computers

A hybrid computer is the combination of properties of both analog and digital computers. These computers consist partly of the components of a digital computer and partly of an analog computer to combine the characteristics of both the types (for example, speed of analog computers and memory & accuracy of digital computers).

SUMMARY

1. The first recognized calculating device was abacus.
2. Napier's Bones and the slide rule were the next calculating devices developed in early 17th century.
3. The first mechanical calculator was developed by Blaise Pascal.
4. Joseph Jacquard introduced punched cards for the power looms.
5. In 1823, Charles Babbage constructed a mechanical machine, namely, "difference engine".
6. He planned another machine "Analytical Engine", but he could not converted his dreams into reality himself, because the standard of engineering technology was not so high at that time.

7. In 1890, Herman Hollerith for the first time used punched cards in his machine.
8. ENIAC stands for Electronic Numerical Integrator And Calculator, developed by John Mauchly and J.P. Eckert in 1946.
9. In 1946, John Neumann brought a revolution in computer technology by introducing stored program concept in EDVAC.
10. The first commercial computer was UNIVAC-1.
11. Growth of the electronic computers spread over the four periods known as four generations.
12. Vacuum tubes were used in first generation computers.
13. ENIAC, EDVAC and UNIVAC-1 were the first generation computers.
14. Transistors were used in 2nd generation computers.
15. The third generation computers used Integrated Circuits (ICs).
16. The fourth generation computers used microprocessor.
17. Computers can be classified into three types:
    i) Analog Computer ii) Digital Computer
    iii) Hybrid Computer

EXERCISE

1. Match column A with column B.

A	B
Howard H. Aiken	Mechanical Calculating Machine
Charles Babbage	Mark - 1
John von Neumann	Analytical Engine
Herman Hollerith	ENIAC
Blaise Pascal	Punched Cards
John Mauchley	EDVAC

2. What is a computer? Briefly trace its history of development.
3. Describe the contribution of Charles Babbage in the development of a computer why Babbage is considered as father of modern computer.
4. Differentiate between digital and analog computer.
5. What is the difference in the first generation computers and the second generation computers.
6. Discuss third and the fourth generation computers.

Chapter 2

MICROCOMPUTERS

2.0 Introduction

Microcomputer (Fig. 2.1) is the smallest in range having very small memory and processing capabilities. It is the microcomputer that has brought computers into schools, colleges, small businesses and home. Microcomputers are also known as personal computers.

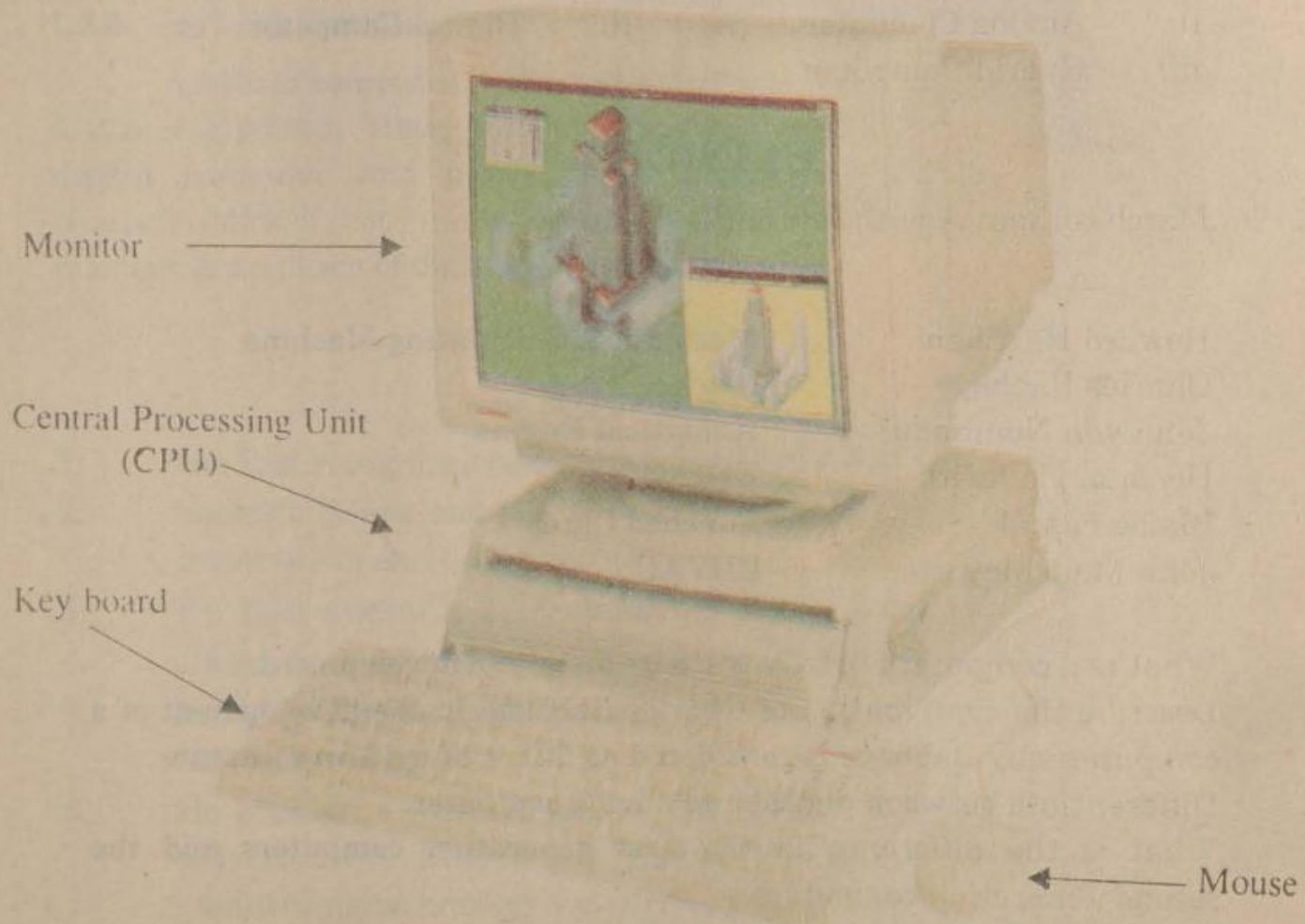


Fig. 2.1 (Microcomputer)

A microcomputer consists of several components, each containing mechanical and electronic equipments. There are three basic components of a microcomputer: Input Unit, Central Processing Unit (CPU) and Output Unit.

2.1 Input Unit

The input unit is like the 'eyes' and 'ears' of the computer. The function of input unit is to feed data or information and a set of instructions into the computer. The most common input devices are the keyboard and the mouse (Fig. 2.1).

2.2 Central Processing Unit (CPU)

The CPU is like the 'brain' of the computer. It controls and supervises all the units in a computer. It is here that the actual work is done. It takes information from the input unit and from the memory, and uses it according to the instructions already given to the computer. From functional point of view CPU consists of three parts, which are:

i) Arithmetic and Logic Unit (ALU) ii) Control Unit (CU)
iii) Memory Unit

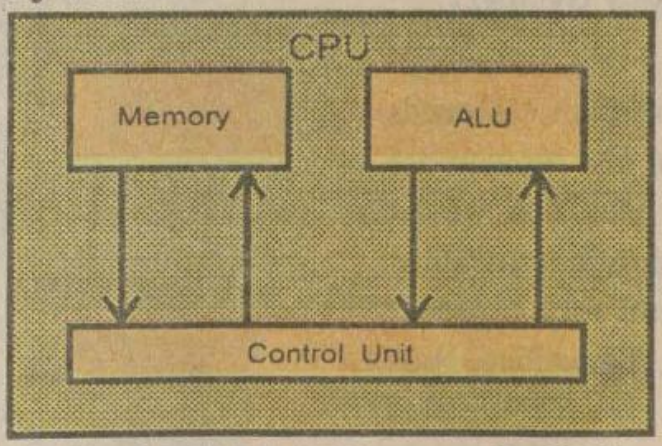


Fig. 2.2 (A Block Diagram of a CPU)

2.2.1 Arithmetic and Logic Unit (ALU)

This part of CPU is capable of performing arithmetic and logic operations. It performs the arithmetic operations such as addition, subtraction, multiplication and division. It can also perform logic operations like comparing two numbers to see which number is greater than, less than or equal to another number.

The ALU performs millions of calculations and comparisons at a very high speed.

2.2.2 Control Unit (CU)

This part of CPU monitors and coordinates all the activities of a complete computer system. It is like a traffic policeman controlling the movements of vehicles on the road. It does not execute the instructions itself, it directs other parts of the computer system to do so. In simple words, it controls the flow of data inside the computer.

2.2.3 Memory Unit

This part of CPU stores data for using it again and again whenever it is required. A computer performs operations on program instruction and data stored in the main memory. The main memory is also called primary storage, internal storage or simply memory.

The memory of a computer has two basic parts:

1. Read Only Memory (ROM)
2. Random Access Memory (RAM)

1. Read Only Memory (ROM) :

This part of memory contains permanently stored informations. When the power is switched off, ROM does not wash away. This information is available to a computer to read and process but not to change it. Therefore, the information which is frequently required for a computer and need not to be changed, is kept on ROM. This information is stored on small pieces of memory chips, before the computer is assembled.

2. Random Access Memory (RAM):

This part of memory consists of blank chips and hence

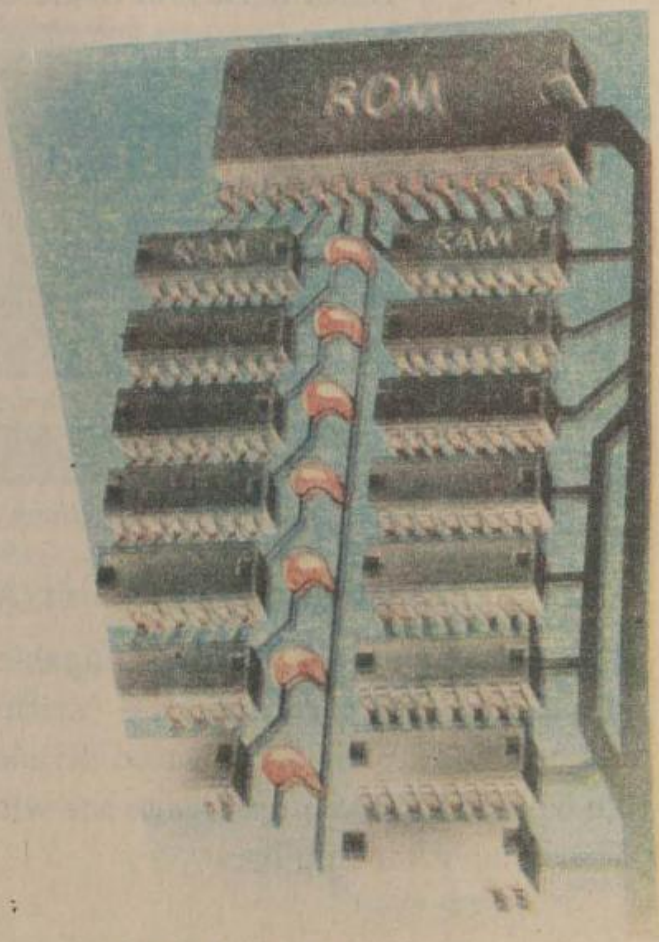


Fig.2.2 (ROM & RAM)

the computer can use it to store and retrieve (write and read) information during its processing. The information stored in RAM is volatile, that is, the stored information is lost, when the computer is shut down.

2.2.4 Units of Memory

A computer memory consists of electronic device which can have one of the two states, called "ON" and "OFF". Numerically, an "ON" is represented by a "1" and "OFF" represented by "0". Either of these binary digits "0" or "1" is called a bit short for *bi*nary digi*t*.

For characters and other information processing, it is appropriate to define a bigger unit instead of a bit. A collection of eight bits is defined as a byte. In memory a byte can store one character. A computer memory is measured in terms of bytes. The larger units of memory are defined as:

1	Kilobyte (1 KB)	=	2^{10} bytes	= 1024 bytes
1	Megabyte (1 MB)	=	1024 KB	
1	Gigabyte (1 GB)	=	1024 MB	

2.3 Output Unit

Output unit provides communication between a computer and the outer world. An output device can be used to display or print the information or final results of computer processing. A variety of output units are used with the computers. The use of these devices depends on the type and purpose of output. Some types of output units are:

* Printers.
* Monitor
* Hard Disk
* Plotters.
* Magnetic Tape
* Floppy Disk

etc. etc.

SUMMARY

1. Microcomputer has three basic components, input unit, CPU and output unit.
2. Keyboard is the most commonly used input device.

3. Arithmetic & Logic Unit (ALU) performs arithmetic and logical operations.
4. Control unit (CU) controls the overall communications and makes the computer to carry out program instructions in the correct order.
5. Memory unit stores information.
6. RAM can store programs and data which are used by the computer. When the power is switched off, it is erased.
7. ROM can only be read and not written. Information is not washed away if the power is switched off.
8. Unit of memory is byte.
9. Information enters the computer through input unit and is then processed by CPU and is available on output unit.
10. Monitors and printers are most commonly used output devices.

EXERCISE

1. What are the functional units of a computer?
2. Describe a CPU and its different units.
3. i) What are the functions of Arithmetic Logic Unit?

   ii) What is the role of Control Unit (CU)?
4. Define memory unit. Discuss its two major types.
5. Define the following:

   i) Bit
   ii) Byte
   iii) Kbyte
   iv) Mbyte
6. Match the correct pairs.

Input	Monitors
CPU	Adds and Compares
Output	Keyboard
ALU	Brain of Computer

INPUT/OUTPUT AND BACKING STORAGE DEVICES

3.0 Introduction

In previous chapter, we studied the functional components of a computer. Now, we shall discus the devices which are used for transferring information between external environment and memory. These devices may be classified into three categories

i) Input devices ii) Output devices

iii) Backing storage devices

3.1 Input Devices

Input is of two types: Data or basic information, and a set of instructions (algorithm) to manipulate this data to derive the required results. The devices used to feed information and algorithm into a computer are called input devices.

Some of the most commonly used input devices are discussed below:

3.1.1 Keyboard

A keyboard is used to feed information and algorithm directly into a computer.

It is just like a typewriter. However, it has several extra keys for specific purposes.

It is comparatively slower in speed, however, its use is very common.

The function of several keys of a keyboard are discussed below.

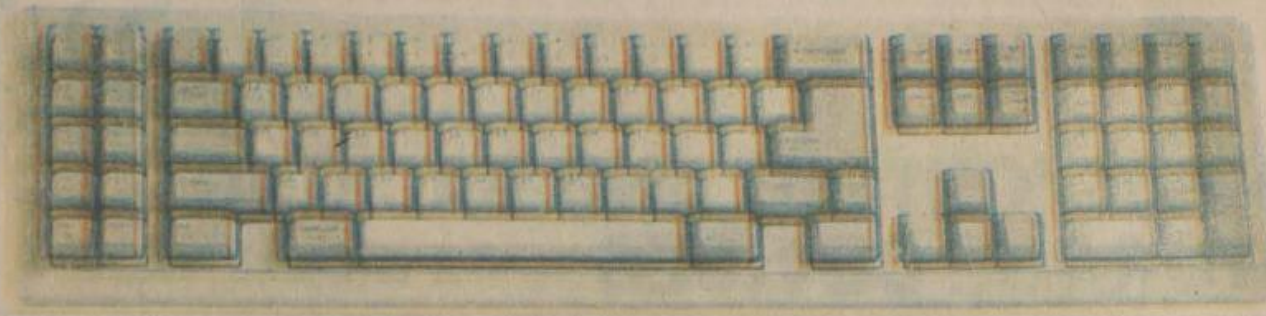


Fig. 3.1 (A Keyboard)

i). ***Standard alphanumeric keys:*** These keys (English Alphabets and special characters) are similar to a standard typewriter and are used for the input of general information.

ii). ***Numeric Keypad:*** These keys are used to input numeric data only. These are very useful in case of large numeric data input because all numeric keys can be accessed by one hand. These keys can also be used as an alternative to screen navigation and editing keys.

iii). ***Function Keys:*** These are keys marked as F1.., F10 or F12 located at the top or left of the keyboard. These are special function keys provided to a programmer to attach special functions to each key.

iv). ***Screen Navigation and Editing Keys:*** These keys are provided to move cursor around in the screen. Many programs use these keys to let the user move around the screen displayed. In some keyboards these keys are provided inside the numeric keypad as alternative keys.

3.1.2 Mouse

A mouse is about the size of a tape cassette. It usually moves on a small bearing and has two or three buttons on the top of it. When a user rolls the mouse across a flat surface, the movement guides a pointer on the screen. It moves in the direction of the movement of the mouse. As the cursor on the screen gives a direct response to the way, the user's hand move the mouse, the user has a sense of

Fig.3.2 (A Mouse)

pointing an object on the screen. Positioning a cursor with a mouse is easier for non-typists than pressing various key combinations.

3.1.3 Joy Stick

It is a lever device that can move in many directions e.g., left, right, forward, backward, etc. It is used to control the movement of an object on the screen. It is mostly used in games.

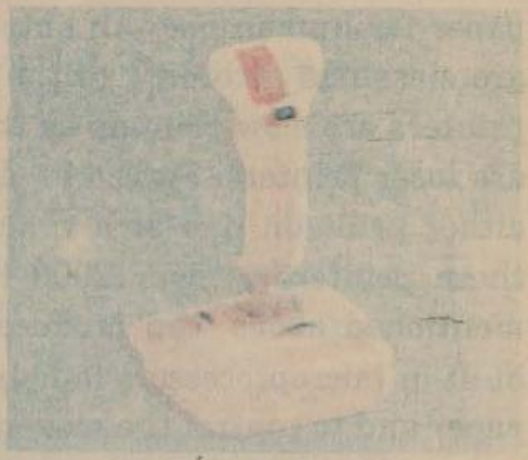


Fig.3.3 (Joy Stick)

3.2 Output Devices

An output device is used by the computer to communicate its results or messages to the outer world. Some of the most commonly used output devices are discussed below:

3.2.1 Monitor

A user would normally like to see the output on the screen of a monitor. A Monitor is the most videly used output device. It is also known as Visual Display Unit (VDU) or Cathode Ray Tube (CRT). It looks like the screen of a television set. It is used to display the output of a computer. The output may be a message telling the user to perform some action or the desired result the user is waiting for. The output sent to a monitor is always in human readable form. Figure 3.4 shows a Monitor.

Fig.3.4 (Monitor)

There are two types of VDUs i.e., those for displaying textual data and those for displaying graphics. VDUs are also classified according to whether they are high resolution or low resolution i.e., how finally they can display graphic or text.

Monitors are available in two major types:

i) Monochrome
ii) Color (CGA, EGA, VGA, Super VGA etc.)

3.2.2 Printers

Printers are output devices used to prepare permanent documents on paper for human use. An output taken on paper is called hardcopy. Printers are classified according to the speed. Dot matrix (Fig 3.4 a) and daisy wheel printers are slow printers as they print one character at a time. Fast printers are laser printers (Fig 3.4 b), drum printers and page printers. These printers either print on line at a time or one page at a time. The average speed of these printers is over 2000 lines per minute. Different types of printers, mentioned above can produce black or multicolor print-outs. Printers have built-in microprocessors to help control the printer-head movement across the paper and to control the movement of the paper up and down.

Fig.3.4a. (Dot Matrix Printer)

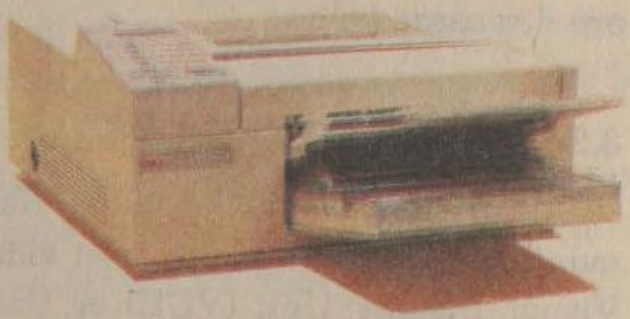


Fig.3.4b. (Laser Printer)

3.3 Backing - Storage Devices

In computer, the data or information has to be stored in an area from where it could be retrieved quickly and correctly. The ideal storage place for this huge amount of data would have been the computer memory. However, computer memory despite being very large, is rather too much limited for this purpose. Further, it is impossible for a computer to retain information in its memory permanently. This information, therefore, is stored on external devices. This storage is called backing storage and the devices used for this purpose are called backing storage devices or secondary storage devices or auxiliary storage devices. These devices are used to provide cheaper and

relatively permanent storage of information. There are a number of such devices available. The commonly used devices are:

i) Cassettes ii) Magnetic Tape iii) Magnetic Disk

i) Cassettes

Some computers use standard audio cassettes. Cassettes are small in size, changeable and inexpensive storage medium for small systems. These are used in the familiar home recorder. Cassettes can store and read information sequentially.

ii) Magnetic Tape

Magnetic tape is a flexible mylar plastic tape, coated on one side with magnetic material. It is widely used when large amount of data are to be processed sequentially. An advantage of magnetic tape is that, it is very cheap and economical storage medium. Its major disadvantage is that, it is very slow and it is only sequential.

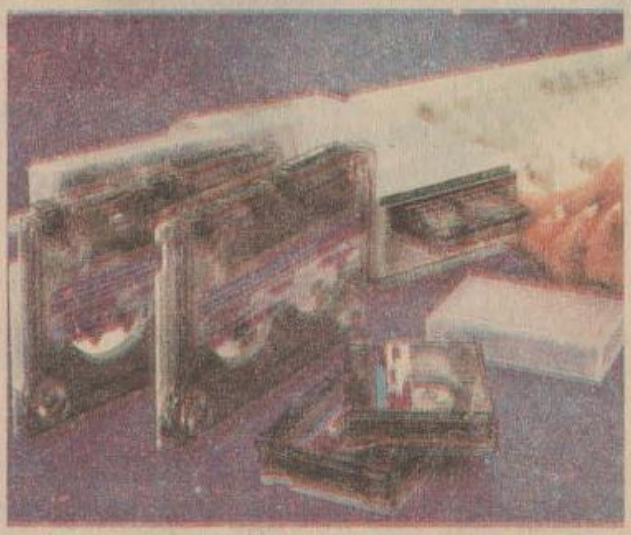


Fig.3.5 (Cassettes & magnetic tapes)

iii) Magnetic Disk

The Magnetic Disk is made up of one or more rotating platters, on which data is stored magnetically. It is made up of plastic, coated on both sides with a special magnetic material. A characteristic of these devices is that a direct access to any item of data can be made without having to read through a sequence of other, irrelevant items.

The magnetic disks are mainly of two types:

i) Hard Disk. ii) Floppy Disk or Diskettes.

Hard disks provide a very big storage capacity and are very fast but these are not interchangeable.

Floppy disks are smaller than the hard disks, because they always consists of only one platter. Floppies are available in sizes of 3.5" and 5.25" inches as shown in the following figure:

Fig.3.6a. (A Hard Disk)

Fig.3.6b. (Floppy Diskettes)

Both of them are available in double density and high density storage capacities. The storage capacities of these diskettes used for IBM compatible computers are given in table 3.1. Floppy disks, although slower than a hard disk, are inexpensive and can easily be moved from computer to computer. However, they are comparatively fragile and must be handled with care.

Table 3.1 (Storage Capacities of Different Diskettes)

Size	Density	Capacity
5.25"	Double	360 KB
5.25"	High	1.2 MB
3.5"	Double	720 KB
3.5"	High	1.44 MB

SUMMARY

1. The devices through which information is transferred into a computer are called input devices, e.g., a keyboard.
2. Devices which enable a computer to transfer information to humans or other machines are called output devices.
3. Printers and monitors are important output devices.
4. Backing storage devices are magnetic tapes, floppy disks, hard disk etc.

EXERCISE

1. List input devices and describe their functions.
2. What is meant by backing storage? Discuss different storages media.
3. How many output devices do you know? Give their brief description.
4. Match the correct pairs.

A	B
Input Device	Backing-storage Device
Output Device	Joy Stick
Diskettes	Visual Display Unit
Memory	Hard Disk

INTRODUCTION TO COMPUTER SOFTWARE

4.0 Introduction

Software is a collection of programs which are of two categories:

i). System Software ii). Application Software

System software is a set of programs which are designed by the manufacturers of computers. These programs are developed for efficient management of hardware. They improve the functional capabilities of the computer and assist the users to solve their problems easily.

Application software is a collection of programs which are written by the users themselves to solve their specific problems. These programs are written in computer languages such as FORTRAN, BASIC and Pascal. Details relating to programming languages, application software and system software are discussed in this chapter.

4.1 Data

Data is a collection of facts and figures. In other words, data is a group of characters (digits, alphabets and/or characters) which represents a specific value or condition. Data provides the foundation for the building blocks of information. For example, the number 3, 33, 7, 100 and 120 are data items.

Some other examples of data items are:

3, 5, 36, 99, 20, 07

A1, B13, F16, K567, AB-10

ISLAMABAD, LAHORE, PESHAWAR, QUETTA, SAEED, HAJRA, WAQAS, JAWAD.

4.2 Information

The organization of data in a meaningful form is called information. In simple words, information is the organized form of data. Note that an information may be considered as data for further processing to get the updated and new version of information. It is exampled in figure 4.1.

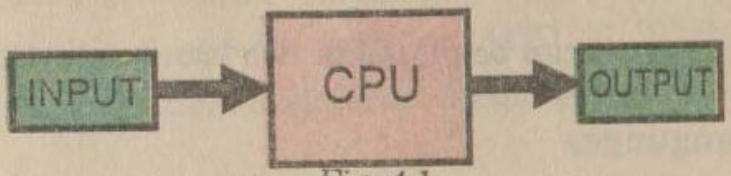


Fig. 4.1

For example as shown above, the data is sent as input to the processor, the results are obtained on the output device. The results are analyzed by the user and this information is again sent as data, to be processed to get further information. This cycle continues till a satisfactory/required result is obtained.

4.3 Types of Data

Data could be of different types depending upon their use and requirements within the processing concerned. Some common types of data are given below:

4.3.1 Numeric Data

The data in the form of numbers is a numeric data, e.g., 1, 2, 300, −20, 30.96 and −5.008 etc.

4.3.2 Alphabetic Data

The data in the form of alphabets which belong to a set of characters e.g.

i) Upper case or capital letters from 'A' to 'Z'.

ii) Lower case or small letters from 'a' to 'z'.

iii) Special symbols e.g., * & " @ etc.

iv) Non-printed characters like tab, space.

4.3.3 Alphanumeric Data

The data in the form of a combination of numerals and alphabets, is an alphanumeric data. e.g., A-1, A123, XYZ4, 5 apples, House-476 etc.

4.4 Computer Languages

A computer language is a means of communication between the user and computer. A computer language defines the rules under which programs are written to make use of computers. There are hundreds of different programming languages available nowadays. Every language has its own set of rules which are used for different operations.

Computer languages can be placed in two broad categories.

4.4.1 Low-level languages

These are:

i) Machine Language
ii) Assembly Language

i) Machine Language

Machine language is the only language that a computer understands directly without any translation. It is the binary language i.e., it is the language of 0s and 1s. It consists of strings of binary numbers.

The binary codes are very difficult to memorize for human beings that is why a machine language is cumbersome for a user.

ii) Assembly Language

An assembly language is one step higher than the machine language in human readability. A machine language consists totally of numbers and is almost impossible for humans to read. In an Assembly language, some of these numbers are replaced by human readable symbols called language for humans and almost of the same efficiency as machine language for computer operations. An assembly language is a combination of memonic operation codes and symbolic codes for addresses.

Assembly language is very efficient but it requires good skills for programming. A program written in an assembly language is translated into a machine language before computer can understand and execute it. A computer program which translates an assembly language program into its equivalent machine language is known as an assembler.

4.4.2 High-level Languages

A language one step higher than low-level languages in human readability is called high-level language. High-level languages are developed to enable the users to program computers easily. These languages are easy to understand. They are also called English-oriented languages in which instructions are given using words, such as ADD, SUBTRACT, INPUT, PRINT, etc. Some commonly used high-level languages are briefly discussed below:

i) FORTRAN

FORTRAN (FORmula TRANslation) was developed in 1957 by a team headed by John Backus. It is the oldest high level language. A full version of FORTRAN was approved by the American National Standard Institute (ANSI) in 1966. It was designed primarily for the use of scientists, engineers and mathematicians for solving mathematical and scientific problems.

ii) BASIC

BASIC (Beginner's All-purpose Symbolic Instruction Code), is one of the most powerful and most widely used high-level language. It was developed in mid 1960s, by John Kemeny and Thomas Kurtz.

Basic language was initially developed to make it convenient for the people to communicate with computers. With the passage of time, millions of people have quickly learned to use BASIC. It has now become one of the major languages of the personal computer users.

iii) COBOL

COBOL (Common Business Oriented Language) was specially designed for business purposes in 1959. It is used all over the world for a variety of data processing tasks, such as payroll, inventory and stock control etc. It is a self-documented language and, it is convenient to understand a program written in COBOL.

iv) PASCAL

Pascal (named after Blaise Pascal, a French mathematician of the seventeenth century) is a general-purpose language which was developed by Professor Niclaus Wirth. Pascal was the first major language developed on the concepts of structure programming.

Pascal can be used for both scientific and business type file processing application. Pascal is the recommended language of learning and teaching in schools and colleges all over the world.

v) C

C is a high-level language, developed in late 1970s. It incorporates many features of both low-level and high-level languages in it. It is used very commonly in system programming. A very popular operating system UNIX is written in this language. It's special features are the efficient use of memory and fast processing speed.

SUMMARY

1. Collection of facts is called data.

2. Data is of three types, numeric, alphabetic and alphanumeric.

3. Numeric data consists of numeric digits or numbers e.g.,

   20 0 −999 +78.55

4. Alphabetic data consists of alphabetic characters e.g.,

   Peshawar, Husnain, Amna, Waleed, Sara.

5. Alphanumeric data consist of numerals, alphabets and special characters e.g., H-9, 267-40-111502, H-NO. 476.

6. There are two types of computer languages, low level languages and high level languages.

7. The language of 0 and I is called Machine language.

8. Assembly language consists of memonic codes.

9. High level languages are like English languages.

EXERCISE

1. Match the following:

Numeric	Used in business
Machine language	Popular language for microcomputer
BASIC	Low level language
COBOL	Type of data

2. Fill in the blanks:

   i) Machine language is a ..... language.

   ii) The physical components of a computer are called .....

   iii) A set of instructions is called .....

   iv) The language used for business application is called .....

   v) FORTRAN stands for .....

3. Differentiate between data and information.

4. What do you mean by a computer languages? Describe its types.

5. Describe briefly any three high level languages.

6. What is the machine language. Discuss its importance.

OPERATING SYSTEMS

5.0 Introduction

An operating system is the collection of programs designed to operate, control and extend the processing capabilities of the computer itself. Unlike application programs, which are written by computer users to meet their own specific information processing needs, operating systems are provided by computer manufacturers.

5.1 Operating System

An operating system is an integrated set of specialized programs that are used to manage the resources and overall operations of a computer. An operating system permits a computer to supervise its own operations by automatically calling the application programs, translating any other special service programs, and managing the data needed to produce the output desired by users.

In the beginning, some operating systems were stored inside the computer's permanent memories. Now-a-days many operating systems are in operation on a variety of computer systems, but some of the most widely used are the Disk Operating Systems (DOS), UNIX and VMS. However, only DOS is discussed in this text.

5.1.1 Disk Operating System

Disk Operating System (DOS) is so called because it resides on the disk of a computer.

An operating system is loaded first in a computer before it can be used for any useful work. Generally, an operating system is loaded from the hard or floppy disk. When the computer is switched on, the supervising portion of the operating system places itself in the RAM of the computer. The supervisor

then loads the different utilities from the DOS on the disk according to the user's requests. When a request from the user is served, that utility is removed from RAM; whereas the supervisor remains in the RAM to serve any new request from the user. If the computer is switched off, the supervisor is washed out from the RAM automatically, but it still remains permanently on the disk. The supervisor is re-loaded when the user wants to use the computer again.

5.1.2 Commonly used Commands of DOS

DOS has its own set of commands. The general syntax of a command is

Command Arguments /Switches

Where command is the name of a DOS command. Arguments is the list of names or character strings required for the command. Switches are the optional characters used to invoke different versions/options of the same command. Some most commonly used commands of DOS, without indicating the prompt symbol, are given below: When the commands are used at the prompt, after each command, ENTER or RETURN key is always pressed.

CLS

This command is used to clear the screen of a monitor.

DIR

This command displays the contents of a directory on a disk. The listing of a directory is composed of 5 columns. The first column is used for the file names. The maximum length of a file name is 8 characters long. The second column designates the file name extension which is of 3 characters at maximum. The third column shows the number of bytes used by a program. The fourth and fifth columns show respectively the date and time, the file was modified last time. The contents of the first and second columns are those at the time of file creation, supplied by the users while the rest of the informations are written by the operating system. When simple DIR command is executed, the entire directory of file can be mentioned through the arguments.

Here the two switches of the DIR command given below:

DIR/P

This command displays the contents of that directory one page at a time.

DIR/W

This command displays all of the contents of the disk without the extensions used with file names on a single screen.

COPY

This command copies a file from one storage area to another storage area of the disk at a time. It does not destroy the original file but only duplicates it.

Examples:

To copy file(s) from one diskette to an other diskette or directory.

a) COPY A:FILE1.EXT B:

This command will copy FILE1 having extension EXT from drive A to drive B with the same name.

b) COPY A:FILE1 B:FILE2

This command will copy FILE1 from drive A to drive B with a new name FILE2.

c) COPY B:*.SYS C:

This command will copy all files represented by * having extension SYS from drive B onto drive C.

VER

Shows the version of the DOS currently in use.

CD

This command changes the directory to the required option given after the "> CD" command.

Examples: CD\HOME\USER

This command will change directory to \HOME\USER

CD..

This command will change the parent directory.

RENAME or REN

This command gives a new name to a file.

Example: RENAME OLDNAME.DAT NEWNAME.DAT

This command will rename the file to NEWNAME.DAT

MKDIR

This command makes a new directory and can also be abbreviated as MD.

Examples: MD DATA

This command will create a new directory DATA in the current directory.

MD\DATA

This command will create a new directory DATA in the root directory.

RMDIR

This command removes the required directory name from the disk. It can also be abbreviated by RD.

Example: RD \DATA

This command will remove the \DATA directory.

DEL

This command deletes files one-by-one from the directory.

Examples: DEL * *

This command will remove all files of the current directory.

DEL TEST.DAT

This command will remove the file TEST.DAT from the current directory.

PATH

This command directs the computer to search a file according to the given path. In the path firstly, we mention the drive letter then the directory/subdirectory name is given if any.

Syntax of Path:

PATH=A:WP\WORK; A:\LOTUS\USER

The above path command given to a computer will be used for searching a file in drive A "WP\WORK" subdirectory. If it fails then it goes to the next step mentioned in a path. The simple PATH command when executed should give the active path.

FORMAT

The format command is used to initialize a disk. Blank diskettes if used cannot store information. For this reason disks have to be initialized.

If we want to format a new floppy placed in Drive A or B, the command at prompt will be given below:

FORMAT A: or FORMAT B:

The FORMAT S/V will format the disk, copy the system files to the disk and ask the user to provide a volume label. (Volume label is discussed later on).

Format is the DOS reserve word followed by the drive letter to be formatted. If the drive letter is not mentioned in the command then the default drive will be considered for initialization. A disk can be formatted to copy system files to the new disk as well or a disk can be formatted and left blank.

The DS,DD disk (360kb) can be formatted in high density drive using the following command:

A > FORMAT B:/4

1) **Format with System**
For this task, we use a switch i.e., "/S" which instructs the computer to initialize a disk in the mentioned/default drive system-wise. The effect of this command will prepare a diskette which will have tracks as well as the system programs which enables the disk to boot/start the computer. The switches are always optional.

2) **Format without System**
In this technique, we need not to specify the "/S" switch, it only prepares the diskette for storing and retrieval of information.

3) **Format with Volume Label**
Volume label is the mark of identification for the disk to be formatted. For this reason we use the "/V" switch, which stands for volume label. When this executes, it asks the user to enter the volume label for which the maximum length is 11 characters. We can also change the current volume label to a new one without formatting a disk. For this reason, we use the "LABEL" command. Sometimes, labels are used for security purpose, especially in case of hard disks.

Other switches are also used with format command to specify the nature of the disk i.e., high density, low density etc.

CHKDSK

It checks the tracks and sectors of a disk for errors and rectifies these. It also displays information about the storage space currently available on the disk and the RAM.

Example: CHKDSK B/F

DISKCOPY

This command copies all of the contents of one disk to another.

Example: DISKCOPY A: B:

5.2 Software Packages

Apart from the usual computer programs, some application programs are developed for different utilities of work ranging from commercial to scientific purposes. These packages are easy to use and fairly quick if the appropriate data processing methods are applied. Some of the commonly used packages for general users are:

1. Word processing
2. Database management
3. Business data analysis or spread sheets

A brief summary of these applications and the facilities provided by different packages is given below:

5.2.1 Word Processing

Word processing is often considered as the "backbone" of applications for office automation. Word processors are also called electronic typewriters. These packages provide the extra benefits of memory for storage, recorrection facility, merging of different documents, setting margins, spell checking, block facilities and so many other functions which are not available in the common typewriters.

Word processors are generally used for the preparation of reports, editing and updating of reports, letters, input and output forms, problem definition, documentation and record keeping purposes. There are different packages available for word processing. The most commonly used packages, at present, are Wordstar and Word perfect.

Other many operations could be performed on a text when it is created. Some of the features are underlining a character or line, bolding

face/highlighting words, centring heading, setting of left and right margins, blocking, superscripts, subscripts and many other operations are performed.

5.2.2 Database Management Systems

These types of packages are used to organize, calculate and draw conclusions from large organizational data. The common examples are the preparation of employee's payroll calculations and stock inventory.

The packages which can be used for this purpose are Dbase, FoxPro, Oracle and Recital etc. Almost all of these packages provide facilities for efficient storage and retrieval of data in files. They can also be used to perform a variety of queries on the stored data and print different reports from it.

The database management systems provide almost all the facilities provided by the high-level language with the enhanced facilities of file handling and report preparation. That is why many people categorise these systems as an evolution of the high-level languages and called them as fourth generation languages (or 4GL).

5.2.3 Business Data Analysis Systems or Spread Sheets

These packages are used in business application. Spread sheets provide a very efficient modes of mathematical calculations on tabular data. Instead of performing calculations on single elements of data they provide facility for performing calculations on the whole column or row of a table to the supplied data.

Spread sheets can also be used to draw conclusions from data and perform a variety of statistical operations. Many spread sheets provide graphical interface and the results can be seen and printed in the form of graphs. Some of the commonly used spread sheets are Lotus123 and QuatroPro.

SUMMARY

1. Operating System is used to manage the resources and overall operations of a computer.
2. DOS command is the abbreviation of Disk Operating System.

3. DIR command displays the contents of a directory and list of files.
4. COPY command copies the file (s) from one storage area to another storage area of the disk.
5. DEL command deletes file (s) from directory and list of files.
6. VER command shows the version of DOS currently in use.
7. MKDIR or MD command makes a new directory.
8. RMDIR or RD command removes the required directory by name from the disk: RM filename.
9. FORMAT command makes the disk in useable form.
10. CHKDSK command shows information about the disk after checking its usability and available space on disk.
11. DISKCOPY command copies all of the contents of one disk to another.
12. Word processors are generally used for the preparation of reports, letters, documentation etc., etc.
13. Database Management System is used to organize, calculate and draw conclusions from large organizational data.
14. Spread Sheets provide a very efficient modes of mathematical calculations on tabular data.

EXERCISE

1. What is an operating system? Describe its importance.
2. What is DOS? Why it is called "Disk Operating System"?
3. Write DOS commands for the following:
   - i) To clear the screen.
   - ii) To display the contents of the directory on a single page.
   - iii) To copy all files of extension BAS from Disk A to B.
   - iv) To copy all files starting with the character S.
   - v) To rename file SUNNY BAS TO KOHALA BAS.
   - vi) To delete all files from B directory.
   - vii) To see the version of DOS.
   - viii) To Format disk with System.
4. Discuss some software packages.

NUMBER SYSTEMS

6.0 Introduction

We have learned in previous chapters that a computer can process only binary numbers 0 and 1. Now we shall learn how a computer converts all other informations into "0" and "1".

The number system which we use in our daily life is called decimal number system because it contains ten digits from 0 to 9.

The total number of digits used in a number system is called its base or radix. Different number systems in common use, and their inter-conversions are discussed in this chapter.

6.1 Number Systems

Some number systems are discussed below:

6.1.1 Decimal Number System

This is a number system which is used in the daily routine work. It consists of ten digits that is from 0 to 9. Therefore base of this system is 10.

6.1.2 Binary Number System

This number system is based on only two digits, the "0" and "1" that is why it is called a binary number system. Therefore base of this number system is 2.

6.1.3 Octal Number System

Octal number system is a short hand version of binary numbers. One octal represents three binary digits. It consists of eight digits i.e., 0, 1, 2, 3, 4, 5, 6, and 7. Therefore base of this number system is 8.

6.1.4 Hexadecimal Number System

Hexadecimal numbers are also a short-hand version of binary numbers. The hexadecimal, also called as Hex, is equivalent to four binary

digits. It comprises of the following sixteen numbers i.e., 0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F. Therefore the base of this system is 16. Here the symbols A,....,F in hex stand respectively for 10,....,15 in decimal. The hex number system uses 16 as its base.

6.2 Inter Conversions of Number Systems

Any number system can be converted into the other number system .

6.2.1 Conversion from Binary to Decimal Number System

Binary numbers are converted into their equivalents in decimal number system by multiplying each bit of binary by its base (i.e.,2) raising to some power. This power starts from n-1, which is the power of the left-most digit, then n-2 and so on. The last power will be zero, n is the total number of bits in the given binary number.

Example 1: Convert (1100) into its decimal equivalent.

$$(1100)_2 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$$
$$= 8 + 4 + 0 + 0$$
$$= 12$$
$$\text{or } (1100)_2 = (12)_{10}$$

Example 2: Convert (10100) into its decimal equivalent.

$$(10100)_2 = 1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$$
$$= 16 + 0 + 4 + 0 + 0$$
$$= 20$$
$$\text{or } (10100)_2 = (20)_{10}$$

Example 3: If there is a period (.) in the binary number e.g. $(1101.01)_2$: The procedure of its conversion is:

$$(1101)_2 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0 \quad | \quad (.01)_2 = 0\times2^{-1} + 1\times2^{-2}$$
$$= 8 + 4 + 0 + 1 \quad | \quad = 0 + 1/4$$
$$= 13 \quad | \quad = 0 + .25$$
$$| \quad = .25$$

Hence $(1101.01)_2 = 13.25$

6.2.2 Conversion from Decimal to Binary Number System

A given decimal number is converted to a binary number by dividing it by the binary base until no more division is possible.

Example 1: Convert $(322)_{10}$ into its binary equivalent.

2	322
2	161-0
2	80-1
2	40-0
2	20-0
2	10-0
2	5-0
2	2-1
2	1-0
	0-1

i.e. $(322)_{10} = (101000010)_2$

Example 2: Convert $(10)_{10}$ into its binary equivalent

2	10
2	5-0
2	2-1
2	1-0
	0-1

i.e. $(10)_{10} = (1010)_2$

6.2.3 Conversion from Decimal Fraction to Binary Fraction

Here the numbers to the right of the decimal point are repeatedly multiplied by 2 till we get a whole number. The product in fraction is assigned as 0-bit whereas in whole number it is assigned as 1-bit.

Example 1: Change $(0.125)_{10}$ into binary fraction.

$0.125 \times 2 = 0.250 = 0$
$0.250 \times 2 = 0.500 = 0$
$0.500 \times 2 = 1.000 = 1$
Hence $(0.125)_{10} = (0.001)_2$

6.2.4 Conversion from Binary to Octal Number System

There are two methods to convert binary numbers to octal numbers

i) Indirect conversion method.

ii) Direct conversion method.

The numbers in binary number system is first converted into decimal equivalent, and then that decimal number is converted to its octal equivalent.

Example 1: Convert $(110111)_2$ into its octal equivalent.

Indirect conversion method:

$$(110111)_2 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$$
$$= 32 + 16 + 0 + 4 + 2 + 1$$
$$= (55)_{10}$$

Now let us divide 55 by 8, the base for octal numbers.

i.e.

8	55
8	6-7
	0-6

$(55)_{10} = (67)_8$

Hence $(110111)_2 = (55)_{10} = (67)_8$

Direct Conversion Method:

The direct method is to group the digits of the given binary number into 3-bits each from the right, and then write the octal equivalent digit of each binary group to get the required octal number.

Example 2: Convert $(110111)_2$ to its octal equivalent by the direct method.

$(110111)_2 = 110 \quad 111$
$= 1\times2^2+1\times2^1+0\times2^0 \quad 1\times2^2+1\times2^1+1\times2^0$
$= 4 + 2 + 0 \quad 4 + 2 + 1$
$= 6 \quad 7$

Hence $(110111)_2 = (67)_8$

It can also be solved by putting the values of binary equivalents 110 and 111 directly from table (6.1) i.e. 6 and 7 respectively.

Table 6.1

Octal Number	Equivalent Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Table 6.2

Hex	Equivalent Binary	Hex	Equivalent Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

6.2.5 Conversion from Binary to Hexadecimal Number System

This is also accomplished in the same manner as for binary to octal conversion.

Example: Convert $(11111)_2$ to equivalent hexadecimal number

$$(11111)_2 = 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$$
$$= 16 + 8 + 4 + 2 + 1 = (31)_{10}$$

Now, divide 31 by 16, the base of the hex to get the required hexa decimal equivalent i.e.,

16	31
16	1-15
	0-1

Which is ·Alternatively, group the number $(11111)_2$ into four bits each starting from right hand side i.e., 0001 1111 and convert each group into its hexadecimal equivalent digit (Table 6.2).

$$0001 = 1 = 1$$
$$1111 = 15 = F$$
$$(11111)_2 = (1F)_{16}$$

6.2.6 Conversion from Hexadecimal to Binary System

This is achieved initially by converting the given hexa into its decimal equivalent, and then come up with the binary by converting that decimal into its binary equivalent.

Example: Convert $(258)_{16}$ into its binary equivalent.

$$(258)_{16} = 2\times16^2 + 5\times16^1 + 8\times16^0$$
$$= 2\times256 + 5\times16 + 8\times1$$
$$= 512 + 80 + 8 = (600)_{10}$$
$$(600)_{10} = (?)_2$$

Now let us divide 600 by 2 to get the binary equivalent i.e.,

2	600
2	300 - 0
2	150 - 0
2	75 - 0
2	37 - 1
2	18 - 1
2	9 - 0
2	4 - 1
2	2 - 0
2	1 - 0
	0 - 1

Which is $(1001011000)_2$. An alternate method is to write the group of 4 binary equivalent digits for each hex digit see table 6.2 given, and then read the resultant bits as a one binary number e.g., $(258)_{16} = (0010\ 0101\ 1000)_2$

SUMMARY

1. Decimal number system consists of ten digits (0 to 9). The base of this system is 10.
2. Binary number system consists of two digits (0,1). The base of this system is 2.
3. Octal number system consists of eight digits (0 to 7). Its base is 8.
4. Hexadecimal number system consists of 16 digits (0 to 9 and A to F) and its base is 16.
5. Computer process data in the form of binary numbers.

EXERCISE

1. What is a number system? Discuss different number systems.
2. Convert the following into binary equivalent:

   i) $(89)_{10}$ ii) $(167)_8$
   iii) $(FA2)_{16}$ iv) $(1F)_{16}$

3. Convert the following into decimal equivalent:

   i) $(111\ 000)_2$ ii) $(1010.011)_2$
   iii) $(123)_8$ iv) $(1.0011)_2$

4. Convert the following into hex equivalent:

   i) $(786)_{10}$ ii) $(1110)_2$
   iii) $(947)_{10}$ iv) $(FB4)_{16}$

PROBLEM-SOLVING AND FLOW CHARTING

7.0 Introduction

A computer cannot solve a problem itself. In order to solve a problem, computers need a set of instructions, called algorithm. An algorithm represents the logic of processing to be performed.

There are two basic requirements to develop an algorithm.

1) The instructions should be clear
2) The desired result should be output in a finite number of steps.

7.1 Problem Solving Process

All problems are solved through the following steps:

Step-1: Define the problem

The first step in solving a problem is to understand and define that problem i.e., what is the purpose of the program? What data is to be processed? What output is required? What is the input?

Step-2: Solve the Problem

Keeping the algorithm in mind, a program is written to solve a problem.

Because an algorithm enables the computer to solve a problem, so algorithm should be designed carefully.

Step-3: Drawing Flowchart

Flowchart is nothing, but a pictorial representation of an algorithm. It is better to meet any problem with the help of a flowcharts.

Step-4: Program Coding

Program coding is the process of writing instructions in the appropriate programming language. Once the problem is well defined then an algorithm represented by a flowchart is quite easily converted into an appropriate programming language.

Step-5: Testing and Debugging

Testing and Debugging are the vital steps in developing computer programs. Testing is the process of making sure the program, performs and desired and debugging is the processing of locating and eliminating the errors.

Step-6: Documentation

Documentation is the process of recording details concerning a computer program. It usually contains many comments lines and many descriptions of how the program works. This documentation can be used in future to modify the program or remove errors.

7.2 Program Flowcharts

Like a road map, program flowcharts are used to reveal how each program is to be developed. It is graphical representation of a problem to be programmed for the machines to produce the required output. It is used to define the problem graphically, before a computer program is written for it.

Program Flowchart helps the programers and system analysts to design and document computer programs. It provides the means of communication and gives a pictorial representation of an algorithm that uses a set of symbols of different shapes to denote different types of instructions. It is used as designing tool and therefore, should be done before the program is developed.

7.2.1 Program Flowchart Symbols

To draw flowchart symbols, "flowcharting template" is usually used which contains a set of symbols of different shapes needed in developing flowcharts. Some of the commonly used program flowcharting symbols are discussed on the following pages:

i) Terminal Symbol:

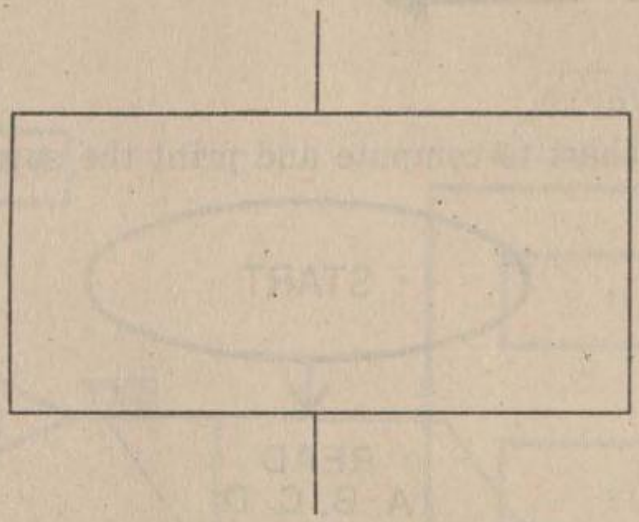
The terminal symbol, as the name implies, is used to indicate a starting or stopping point in a flowchart.

1. Begin a flowchart with

2. End it with

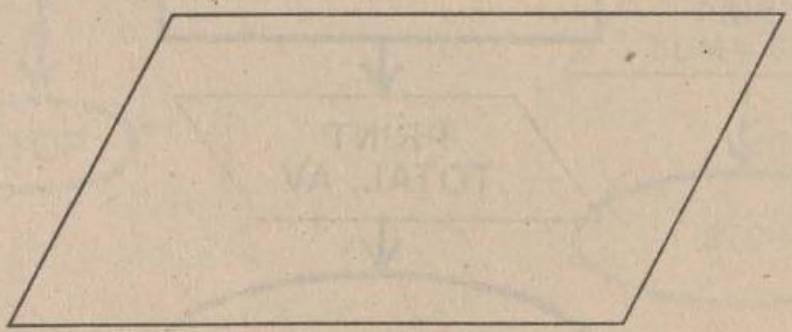
ii) Processing Symbol

It is used to indicate any type of arithmetic calculations.



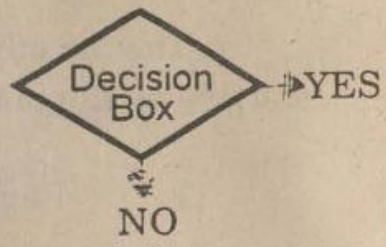
iii) Input/output Symbol

It is used to indicate any input/output



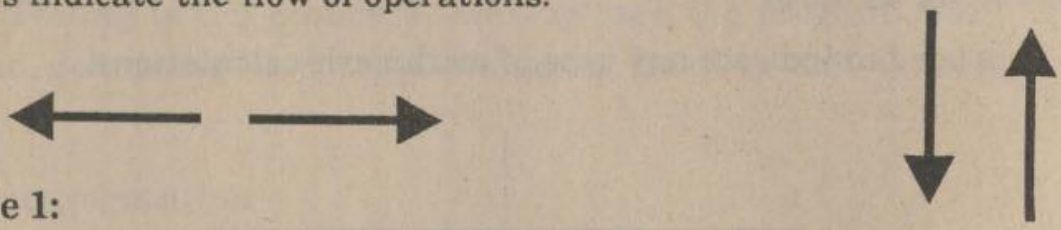
iv) Decision Symbol

One of the important flowcharting symbols is the decision. It reveals what decision or action the computer is to be taken as a result of the condition.



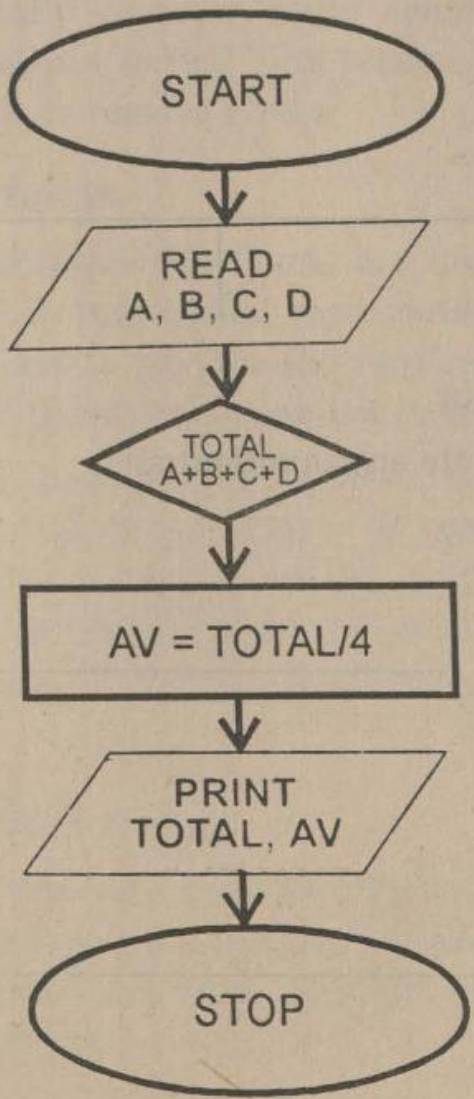
v) Flowlines

Flowcharting symbols are connected together by means of flowlines. Flowlines indicate the flow of operations.



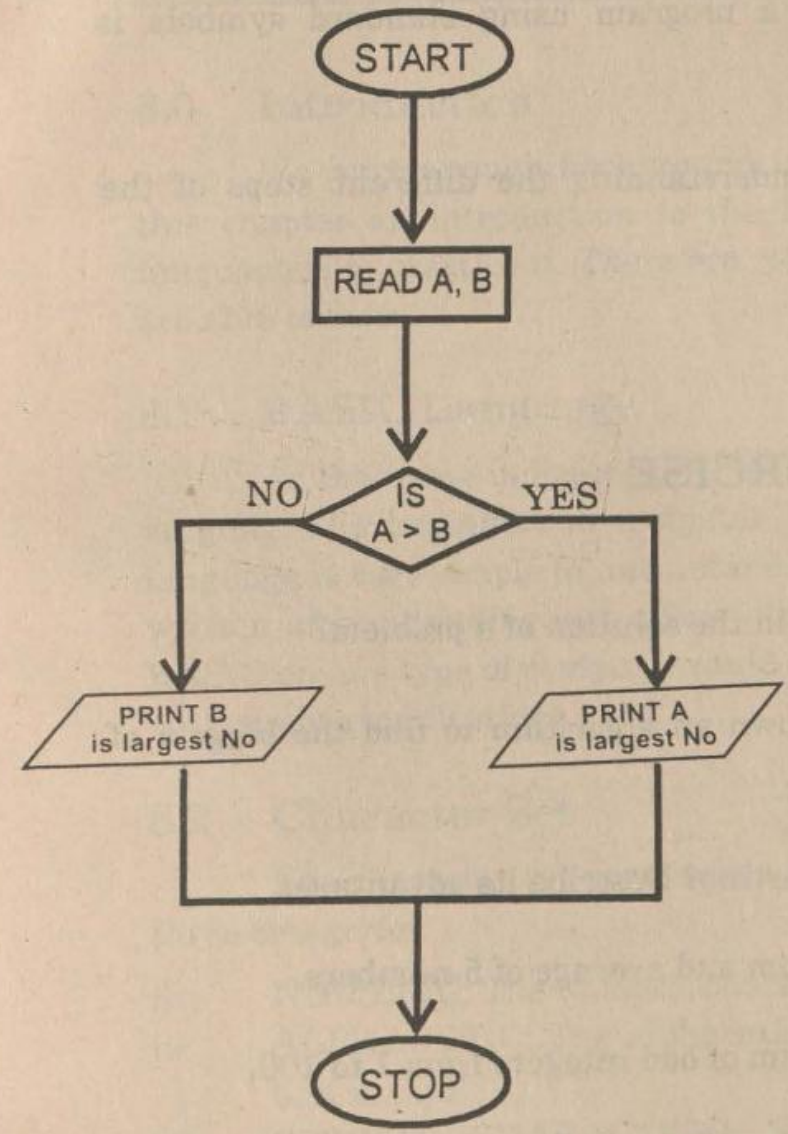
Example 1:

Draw a flowchart to compute and print the sum and average for the given four numbers.



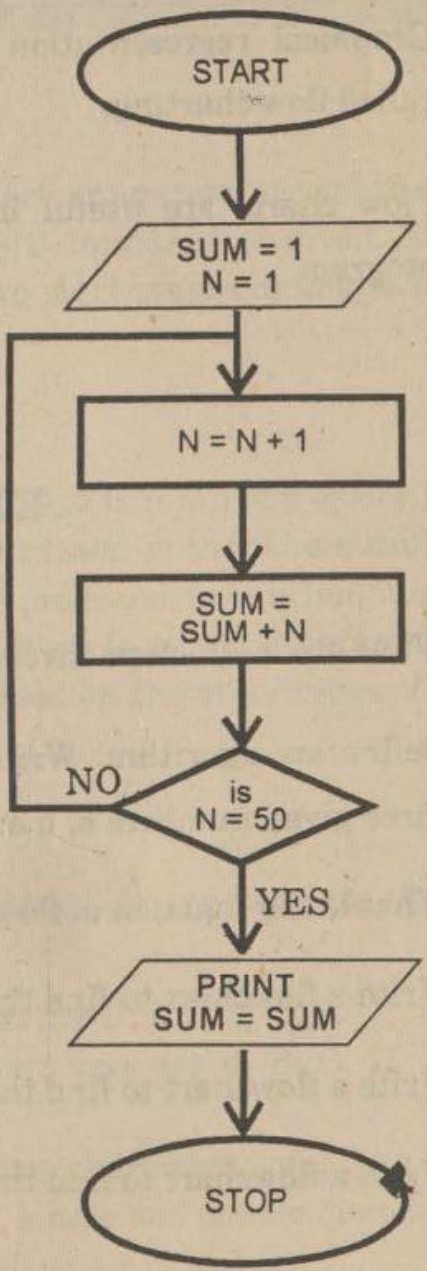
Example 2:

Read any two numbers, Draw a flowchart which finds the largest number.



Example 3:

Draw a flowchart to find out the total sum of first 50 natural numbers.



SUMMARY

1. Solution of problems is sequentially broken down into detailed steps.
2. Minimum six steps are required in the problem solving process.
3. Graphical representation of a program using standard symbols is called flow charting.
4. Flow charts are useful in understanding the different steps of the program.

EXERCISE

1. What are main steps involved in the solution of a problem?
2. Define an algorithm. Write down an algorithm to find the biggest of three given numbers a, b and c.
3. What is the function of flowcharting? Describe its advantages.
4. Write a flowchart to find the sum and average of 5 numbers.
5. Write a flowchart to find the sum of odd integers from 1 to 100.
6. Write a flowchart to find the area of a triangle.

PROGRAMMING IN BASIC LANGUAGE

8.0 Introduction

We have enough background to start programming our computer. In this chapter an introduction to the BASIC language is given. To learn a language is to practice it. Therefore, we can start programming as soon as we are able to do so.

8.1 BASIC Language

As the name indicates, BASIC language is relatively easier than other languages for beginners in computer. The reason is that the structure of the language is very simple to understand and programs in this language could be written after attending just a few introductory lectures. Programs written in BASIC on one type of computer could be used on the other types of computers with slight modifications.

8.2 Character Set

The character set permissible in BASIC is grouped into the following three categories:

i) NUMERIC: The numeric characters are 0, 1, 2, 3, ...,9.

ii) ALPHABETIC: The alphabetic characters are A, B, C, D, ...,Z and a, b, c, d,...., z.

iii) SPECIAL CHARACTERS: Special characters are: the period (.), comma (,), semicolons (;), colons (:), single and double quotes ('), ("), plus and minus (+, −), asterisk (*), slash (/) ,(=, < >) etc.

The blank character is equivalent to a space on the screen. Some versions of BASIC use other special characters as well.

8.2.1 Numeric Constants

A numeric constant is a simple fixed number that can be assigned to a variable in a program. Some of the valid numeric constants are as follows:

786 -8.56 78549 +0.44 -25 etc.

8.2.2 String Constants

String constant is a non-numeric value that can only be assigned to the string variable discussed later. In string constants, we can write any combination of characters enclosed in quotes. For example "ALI", "AYSHA" "PAKISTAN", "042-5758561" etc.

8.3 Variables

As opposed to a constant, the value of a variable does not remain fixed. In every program written in BASIC, variable names need to be used since the values associated with a variable name can be different at different times.

The two types of variables are discussed below:

8.3.1 Numeric Variables

The variables associated only with numbers are called numeric variables.

8.3.2 String Variables

The variables associated with the characters (i.e, letters, numbers, special characters) are known as string variables.

Rules for Numeric & String Variables:

1) We can use any single letter.
   e.g., A, X, P, Q, Z etc.
2) We can also make combination of letters.
   e.g., ABC, SSM, JAVAID, FATIMA, ALI etc.
3) Combination of letters and numbers are also allowed but first character must be an English alphabet.
   e.g., A121, H85, F16, PTV2 etc.
4) Space (blank) is not allowed.
5) To differentiate between numeric and string variable names, we always attach $ (Dollar sign) at the end of the variable name.

e.g., PAK$, STOCK$, A$, ABC$ etc.

6) Length of the variable name varies from one BASIC interpreter/compiler to another. However in GWBASIC first 40 characters are significant.

8.4 Reserved Words

Reserved words are those words which can be useful for specific purposes. These words should not be used as variable names. There are different reserve words e.g. READ, PRINT, INPUT, IF, END, etc. The list of reserved name is given in appendix A.

8.5 Expressions

An expression is a combination of constants and variables linked by arithmetic operators like +, −, *, /. Expressions are used to perform arithmetic operations. The expressions are evaluated from left to right, but some operations have priority over the others. These are discussed below:

Parentheses are evaluated first, second exponential then multiplication and division have the same priority from left to right. Similarly, the addition and subtraction are evaluated with equal priorities. If all the operators used in an expression have the same priority, then the expression will be executed from left to right.

The expressions can be of three types:

i) Arithmetic expression.

ii) Relational expression.

iii) Logical expression.

Details relating to relational and logical expressions are provided in the chapter on transfer of control statements.

8.5.1 Arithmetic Expression

In an arithmetic expression, the following arithmetic operators are used in conjunction with the operands.

Table 8.1

Types of Operation	Arithmetic symbol	Symbol in Basic
Addition	+	+
Subtraction	−	−
Division	÷	/
Multiplication	x	*
Exponentiation*	•	** or ^
Equality	=	=
Lesser than	<	<
Greater than	>	>
Lesser than or equal to	≤	<=
Greater than or equal to	≥	>=
Not equal to	≠	<>

Example:

30−2^ 3 + 4 / 2−3* (3 + 2* 2)

STEP - 1 ( ) = 30−2^3 + 4 / 2−3* (3 + 4) = 30−2^3 + 4 / 2−3* (7)

STEP - 2 ^ = 30−8 + 4 / 2−3* 7

STEP - 3 / = 30−8 + 2−3* (7)

STEP - 4 * = 30−8 + 2−21

STEP - 5 − = 22 + 2−21

STEP - 6 + = 24 − 21

STEP - 7 − = 3

8.6 Statements and Statement Number

Programs in BASIC language are written in a sequence which consists of a series of statements. Each statement can be of one or more lines in length. Each statement in a program must start with a statement number. These numbers can go from 1 to 99999 and must be integers. These numbers are used for the following two purposes:

i) The computers use them to assign a memory location for the statement.

ii) The numbers tell the computer about the order of execution of the program statements.

A short program that illustrates the use of statement numbers is shown below:

```
10 REM  BASIC Program To Add Two Numbers.
20 READ  A, B
30 LET C = A + B
40 PRINT  C
50 DATA 2, 3
60 END
```

It should be noted that in this program individual statements are numbered. The first line is numbered 10, the second 20 and so on. They are not numbered 1,2,3,4,5 in the above example. The reason is quite often, that programmers change their mind and decide to insert new statements in the program.

```
10 REM BASIC PROGRAM TO ADD TWO NUMBERS
15 REM DATE WRITTEN 12TH, OCTOBER 1995.
20 READ A, B
30 LET  C = A + B
40 PRINT C
50 DATA 2,3
60 END
```

In this program REM statement is being inserted at line 15. Here, it was quite easy to insert a new line number. It will, however, be impossible if there is 1 instead of 10 and 2 instead of 20. The reason is that BASIC does not allow line numbers in fractions.

8.7 Arithmetic Assignment Statement

LET statement in BASIC is referred to as a replacement or an assignment statement. The reason is that, this statement allows variables to accept new values. The general form of the LET statement is:

Statement number LET V = {constant } {variable }
{ arithmetic expression}

The LET statement is written in the following order. The statement number, the reserved word LET, a single variable, an equal sign and any one of the arithmetic expression, constant or variable. In case of an expression, value of that expression is first computed according to the rules of arithmetic operations and the result is assigned to the variable that appears to the left of the "=" sign. It is important to note that only one variable can be assigned a single value at a time, and that the "=" sign does not have the same meaning as in algebra. After the evaluation of the arithmetic expression on the right hand side, computer will assign that specific numerical value to the variable on the left of the "=" sign. Examples of the LET statement are:

```
10 LET X = 5
20 LET Y = A + B
30 LET A = B
40 LET K = K + 1
```

In the above example, at line number 10, X is assigned the value 5. In line 20, the arithmetic expression A + B is evaluated according to the current value of A and B and its value is assigned to the variable name Y. In line 30, the value of B is copied to A. In the last line, the value of K is incremented by 1.

When using the LET statement, it is important to realize that any variable used on the right hand side must be known to the program prior to its use.

In many new versions of BASIC language, for compilers and interpreters, the word LET is optional; that is, the equal sign is sufficient when assigning an expression to a variable name.

SUMMARY

1. Constants are the fixed values. These are of two types: numeric and string.
2. Values of the variables may not remain fixed during the execution. These are of two types: Numeric and String Variable.
3. An arithmetic expression is a combination of constants and variables linked by arithmetic operator (s).
4. Statement number must be unsigned positive integer.
5. LET statement is the assignment statement.

EXERCISE

1. Fill in the blanks:
   i) String constant must be enclosed in _____.
   ii) An expression is a combination of _____.
   iii) The major function of a statement number is _____.
   iv) A _____ is a numeric value that does not change.
2. Define a variable and briefly explain its different types used in BASIC language.
3. Define a constant and its types.
4. How would you define a BASIC arithmetic expression? Discuss the arithmetic operators used in BASIC arithmetic expressions.
5. Compute the following expression.
   i) 3 + 9 ^ 2 * 5 - 1 ii) 10 - 5* 16 / 8 + 6 iii) 6 / 3 * 5 ^ 3 - 7

INPUT AND OUTPUT STATEMENTS

9.0 Introduction

Many programs require data during their execution. It is nice if the computer requests for the data. While the program is executed, data may be read into the variables. INPUT and READ/DATA statements are used for this purpose. In order to get the results, output statement e.g., PRINT and LPRINT are used. These statements are discussed in this chapter.

9.1 The INPUT Statement

Input statement is used to accept numeric or string data from the keyboard and assign this data to the variables used within the INPUT statement during the program execution.

Syntax:

LINE TO INPUT "MESSAGE" < VARIABLE NAMES >

The syntax shows a line number followed by the reserved word INPUT followed by a list of variables separated by commas. Examples are:

```
10 INPUT A
20 INPUT TEMP, HEAT
```

When the above segment of BASIC program is run on a computer, the program will first wait for the value of "A" i.e., some numeric value should be entered during the execution for the numeric variable A; whereas the second line will accept two numeric values, one for TEPM and the other for HEAT.

In the above segment, the program will wait for the values to be entered without displaying any message for it. This can make the situation a bit confusing and sometimes even complicated when more than one variables are used in a single INPUT statement. To avoid this difficulty, we can add a comment to each INPUT statement clearly.

Syntax:

Line No. INPUT string, variable

Example:

```
10 INPUT "Please enter your name", N$
20 INPUT "Please enter your age", AG
```

The above examples will display the message asking for the name to be keyed-in and will wait until it is provided by the user. When a value is entered for the name, the next statement will display the message asking for the age to be keyed-in and then wait for the input from the user.

Zero will be assigned to the variables. In some variables of BASIC "out of data" error message would be displayed on the screen. If it is the other round i.e. if the values are more than the variables, then the extra values will be ignored.

9.2 The READ/DATA Statement

READ statement is used to assign numeric and non-numeric values to the variables. These values are supplied in DATA statement. There should be one to one correspondence between variables of the READ statement and constant values of the DATA statement.

Syntax:

```
Line No. READ list of variables
Line No. DATA list of literals
```

Example:

```
10 READ X,Y,Z
20 DATA 10, 20, 30
```

In the above example, X will be assigned the value 10, Y will be assigned the value 20 and Z the value 30. The effect of the above example will be the same as:

```
10 X = 10
15 Y = 20
20 Z = 30
```

DATA statement is used only with a READ statement. If a READ statement contains more variables than constants in the corresponding DATA statement, zero will be assigned to the variable. If it is the other way round, then the extra constants will be ignored.

Two types of variables are used in READ statement, one is a numeric variable for which the numeric data is used in DATA statement; whereas the other data type is string where the corresponding value used in the DATA statement must be of the string type.

Example 1:

```
10 READ A, B$
20 DATA 9, "ARSHAD"
```

In the above example, A is a numeric variable and the numeric value of 9 will be assigned to it from the DATA statement, while B$ is a string variable and the non-numeric value i.e., "ARSHAD" will be assigned to it from the DATA statement.

Example 2:

```
10 READ X,Y,Z
20 DATA 5,6
30 READ A,B,C
40 DATA 7,8
50 DATA 9,10
```

Example 3:

```
10 READ X,Y,Z,A,B,C
20 DATA 5,6,7,8,9,10
```

Both of the above examples will produce the same result but in example "2" the first READ statement at line 10 contains more variables that the DATA statement at line 20. Therefore, Z will be assigned value 7 from the DATA statement at line 40. Similarly the READ Statement at line 30 will take its first value from DATA at line 40 (the last remaining value) and the rest of the values from DATA at line 50. The result of this will be:

X=5 Y=6 Z=7 A=8 B=9 C=10

9.2.1 Restore Statement

Restore statement allows data values, in a DATA statement, to be re-used in the program. After RESTORE statement, we can read all the DATA again for the further process.

Example:

```
10 READ A,B
20 RESTORE
30 READ X,Y
```

```
40 PRINT "A=", A
50 PRINT "B=", B
60 PRINT "X=", Y
70 PRINT "Y=", Y
80 DATA 5,6
90 END
RUN
A = 5
B = 6
X = 5
Y = 6
```

9.3 The PRINT Statement

The PRINT statement displays string (s) or the values of variable (s) on the monitor.

Syntax:

Line No. PRINT {string} {variables}

The curly brackets in the syntax show the string or variables which can be used with a PRINT statement.

Example 1:

```
10 LET A = 10
20 PRINT A
```

The above program displays the value of A. i.e., 10 on the screen.

```
10 LET A$ = "AFZAL"
20 LET B$ = "AKHTAR"
30 PRINT A$, B$
```

The above example displays the contents of A$ and B$ i.e., AFZAL and AKHTAR on the screen.

Sometimes, the contents of variable (s) are not meaningful when displayed alone and should therefore be accompanied by a message.

Example 2:

```
10 LET A = 10
20 LET B = 20
30 LET SUM = A + B
```

```
40 PRINT "THE SUM OF TWO NOS. IS", SUM
50 END
```

The example will not only display the contents of the SUM variable which is 30, but will also display the message in the PRINT statement before the value of SUM. This makes the results more understandable.

If PRINT statement does not contain any string or variable then a blank line is displayed on the screen.

Example 3:

```
10 PRINT "Pakistan"
20 PRINT
30 PRINT "Zindabad"
```

The result of the above program will be as follows:

```
Pakistan

Zindabad
```

The blank line is inserted because of the PRINT statement at live no.20.

Example 4:

```
10 LET A = 10
20 LET B = 20
30 LET C = 30
40 PRINT A,B,C
50 PRINT "Pakistan"
60 PRINT "Zindabad"
70 END
```

When the variables in a PRINT statement are separated by commas then the variables will be displayed on the screen in five zones. That is, a single output line will be divided into 5 zones and each variable will occupy a single zone. When a semi-colon is used in the PRINT statement between the variables or strings, then the values will be displayed without spaces between them, however numeric values leave a space between themselves. Using a semicolon at the end of PRINT statement will display the result of the next PRINT statement on the same line.

For example, the output of the above program will be:

```
10          20          30
Pakistan  Zindabad
```

9.4 The LPRINT Statement

The LPRINT statement sends strings and values of variables to a printer. In other words, it directs the output to the printer rather than to a monitor.

Syntax:

Line No. LPRINT {string} {variable}

Example:

```
10 LET A$ = "ASGHAR"
20 LPRINT A$
```

The above program will send the output ASGHAR to the printer.

SUMMARY

1. INPUT statement accepts data directly from the terminal during program execution.
2. DATA may be read into the variables using READ/DATA statements.
3. The PRINT statement displays string (s) or values of variable (s) on the monitor.
4. If a PRINT statement does not contain any string or variable, then a blank line is displayed on the screen.
5. The LPRINT statement sends strings and values of variables to a printer.

EXERCISE

1. Write T for true and F for false for the following:
   a. READ statement is an output statement.
   b. INPUT statement accepts values from disks.
   c. Print A$ + B is a valid BASIC statement.
   d. LPRINT statement sends the data to a printer.
   e. Semicolon at the end of a PRINT statement causes the output of the next PRINT statement to be displayed on the same line.

2. Fill in the blanks with appropriate words:
   a. Comma between variables divides the screen into ..... zones.
   b. INPUT statement accepts data from a .....
   c. DATA statement is used with ..... statement.
   d. The general syntax of input statement is INPUT string.....

3. Find errors in the following program:
   i.
```
10 READ X,Y,Z
20 LET A = X* Y* Z
30 PRINT "THE PRODUCT IS", A
40 END
```
   ii.
```
10 INPUT "ENTER TEMPERATURE", F
20 C = 5/9* (32-F) : PRINT
30 PRINT "THE RESULT IS" C
40 END
```

4. What will be the output of the following program?
   i.
```
10 READ X,Y,Z
20 A = X + Y + Z
30 PRINT A,X,Y,Z
40 DATA 5
50 DATA 6,11
60 END
```
   ii.
```
10 READ X,Y,Z
20 A = X + Y/Z
30 PRINT A
40 PRINT
50 LPRINT "THE RESULT IS", A
60 DATA 20,40,2
70 END
```

5. Write a BASIC program to accept (input) 5 numbers, calculate their sum, product, average and display the results on the screen.

6. Categorize the following into input and output statements.

   READ, PRINT, INPUT, LPRINT

TRANSFER OF CONTROL STATEMENT

10.0 Introduction

This statement branches/tranfers the control from one part of the program to another specified part of the program. Some of the transfers are based on condition and hence known as the conditional transfer of control statements. The unconditional transfer of control statements do not require any condition to be specified in the statement of the program.

The transfer of control statements is also used to perform loops i.e., to repeat part of a program multiple times. The number of times a loop is repeated depends on certain condition. Therefore, the conditional transfer of control is used to produce looping effect.

10.1 The GOTO Statement

The GOTO is a branching statement. It breaks the normal sequential execution and transfers the control from one part of the program to another. This type of transfer of control is unconditional.

Syntax:

Line no. GOTO line no.

Example:

```
10 PRINT "ABC"
20 PRINT "XYZ"
30 GOTO 10
```

After processing the line 10, computer will come to line 20 and 30 but it will get instruction to go back to line 10 instead of going to line 40. So again it will process 10, 20 and 30, and will find the same instruction to go back to line 10 and so on. We shall notice that a closed path will be formed which is called LOOP. This type of programming is called LOOPING or infinite looping (continuous looping). When a computer is in a loop, we don't

have control for supplying any more instructions. For this purpose we have to break the loop manually. We use Ctrl + C or Ctrl + BREAK key to break the loop. To make it continue from the same instruction where it was broken, we use a command CONT which stands for "CONTINUE".

On the other hand if we will transfer the control from one line to another which will not form a loop is known as breaking.

Example:

```
10 PRINT "TEST FOR BRANCHING"
20 GOTO 50
30 PRINT "SKIP-1"
40 PRINT "SKIP-2"
50 PRINT "I HAVE; SKIPPED 2 LINES"
60 PRINT "THAT'S ENOUGH"
70 END
RUN
TEST FOR BRANCHING
I HAVE SKIPPED TWO LINES
THAT'S ENOUGH
```

Note that we can also write GOTO as GO TO branching with simple GOTO is called unconditional branching. We will study CONDITIONAL branching in the next topic.

10.2 The ON GOTO Statement

This statement is used for multiple branching. Unlike GOTO statement, which is used for simple branching i.e., sending control from one part of the program to another. This statement is favoured when there are several alternatives rather than a single one. This statement is normally used in the selection of menu choices. In this case, many choices are controlled by a variable and on the basis of its value, the control is transferred to a particular area of the program.

Syntax:

Line no. ON variable/expression GOTO statement No. 1, No.2, ...

The statement contains the reserved word ON followed by a numeric variable or formula and two or more statement numbers each separated by comma.

Example 1:

```
10 ON CH GOTO  100, 200, 300, 400
```

The above line will transfer control to an appropriate statement depending on the value of variable CH. If CH has the value 1, the control will be transferred to the first statement number i.e., 100. If CH has value 2, control will be transferred to the second statement number i.e., 200 and so on.

Example 2:

```
10 REM USE OF ON / GOTO STATEMENT
20 PRINT "MAIN MENU"
30 PRINT
40 PRINT "1. SUM PROGRAM"
50 PRINT "2. PRODUCT PROGRAM"
60 PRINT "3. AVERAGE PROGRAM"
70 PRINT "4. EXIT"
80 INPUT "ENTER YOUR CHOICE FROM 1-4", CH
90 ON CH GO TO 200, 300, 400, 500
100 GO TO 10
200 REM SUM PROGRAM
210 INPUT "FIRST VALUE", A
220 INPUT "SECOND VALUE", B
230 INPUT "THIRD VALUE", C
240 LET SUM = A + B + C
250 PRINT "THE SUM IS", SUM
260 GO TO 20
300 REM PRODUCT PROGRAM
310 INPUT "FIRST VALUE", A
320 INPUT "SECOND VALUE", B
330 INPUT "THIRD VALUE", C
340 LET PRO = A * B * C
350 PRINT "THE PRODUCT IS", PRO
360 GO TO 20
400 REM AVERAGE PROGRAM
410 INPUT "FIRST VALUE", A
420 INPUT "SECOND VALUE", B
430 INPUT "THIRD VALUE", C
440 LET AVG = (A + B + C)/3
```

```
450 PRINT "THE AVERAGE IS", AVG
460 GOTO 20
500 END
```

10.3 The IF-THEN-ELSE Statement

The IF statement is used for conditional branching. It transfers control to a specific portion of a program depending on the condition. In other words, it transfers control into one of the two specific statements mentioned in the IF-THEN-ELSE statements.

Syntax: Line no. IF < condition > THEN statement -1 ELSE statement-2

The IF statement determines whether the condition is true or false. If it is true then statement-1 will be executed otherwise statement-2 will be executed. The ELSE part of the statement is optional and can be omitted. If the ELSE portion is not used in the IF statement and if the condition is false then the control will be transferred to the next executable statement after the IF statement.

The condition of an IF statement can be a relational expression, a logical expression or a combination of the two. These types of expressions are explained below:

10.4 Relational Expression

A relational expression is composed of operands linked by the relational operators. The relational operators used in the relational expressions are given in table 10.1

Table 10.1

Symbols	Symbols
=	Equal to
>	Greater than
<	Less than
< >	Not equal to
> =	Greater than or equal to
< =	Less than or equal to

Example:

A > B B < > C

The above relational expressions make comparison between variables after which the appropriate action can be taken by some command.

10.5 Logical Expressions

When a selection is based upon one or more conditions being true, it is possible to combine the conditions together using logical operators and the resulting condition would either be true or false. The most commonly used logical operators are AND, OR and NOT.

LOGICAL AND:

Table 10.2 (Truth table for logical and)

Condition X	Condition Y	X and Y
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

As shown in the table 10.2 "X AND Y" is true only if X is true and Y is true.

Example 1: IF (A=B) AND (C=D) THEN LET T = A + B + C + D

The statement following the THEN would be executed by the computer only if A = B is true and C = D is true.

LOGICAL OR:

Table 10.3 (Truth table for logical or)

Condition X	Condition Y	X Or Y
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TURE	FALSE	TRUE
TRUE	TRUE	TRUE

"X Or Y" is true if X is true or Y is true or both are true i.e., if at least one of the conditions is true.

Example 2: IF (A < B) OR (C > D) THEN LET D = A* B* C* D

The statement following THEN will be executed by the computer if either A<B is true or C>D is true or both are true.

LOGICAL NOT:

Table 10.4
(Truth Table for Logical NOT)

Condition X	NOT X
FALSE	TRUE
TRUE	FALSE

The logical operator NOT negates or reverses a condition, so true becomes false and false becomes true.

Example 3: IF NOT (A < B) THEN LET P = S*T

This implies that if A < B is false then NOT (A<B) will be true and the statement after THEN will be executed by the computer.

Example 4:

i)
```
10 REM USE OF IF STATEMENT
20 INPUT "ENTER VALUE OF A", A
30 INPUT "ENTER VALUE OF B", B
40 IF A > B THEN PRINT "THE LARGEST VALUE IS", A ELSE
PRINT "THE LARGEST VALUE IS", B
50 END
```

ii)
```
10 REM USE OF IF STATEMENT
20 INPUT "ENTER FIRST VALUE", A
30 INPUT "ENTER SECOND VALUE", B
40 INPUT "ENTER THIRD VALUE", C
50 IF A > B AND A > C THEN LARGEST = A
60 IF B > A AND B > C THEN LARGEST = B
70 IF C > A AND C > B THEN LARGEST = C
```

```
80 PRINT "THE LARGEST VALUE = ", LARGEST
90 END
```

SUMMARY

1. A transfer that is based on some condition (s) is known as conditional transfer of controls.
2. Unconditional transfer of control do not require any condition.
3. GOTO statement is unconditional transfer of control.
4. ON/GOTO statement is used for multiple branching.
5. IF statement is used for conditional branching.
6. With the help of ELSE option, and action defined in case the condition is false.
7. The relational operators are =, >, <, < >, > = and < =
8. Logical operators are AND, OR, NOT.

EXERCISE

1. What is transfer of control? Discuss statement causing transfer of control.
2. Explain the difference between a GOTO and ON/GOTO statement.
3. What are the six relational operators used in BASIC?
4. Differentiate between IF-THEN and ON-GOTO statement.
5. Indicate; what is wrong (if any):
   i) GOTO M
   ii) GOTO - 100
   iii) IF A > B GOTO PRINT
   iv) If N$ = ALEEZA THEN GOTO 100
   v) IF A = B = C THEN PRINT "Numbers are equal"
   vi) ON 30 GOTO 40, 50, 60, 70
6. Write a BASIC program to INPUT any number and display whether that number is even or odd.
7. Write a program that reads five numbers and print either one of the two statements given below:
   ALL NUMBERS ARE EQUAL.
   or
   ALL NUMBERS ARE NOT EQUAL.

REPETITIVE STATEMENTS

11.0 Introduction

The process of executing the same segment of a program is known as looping. While writing a program, sometimes it is necessary to repeat a part of a program for a number of times. One solution to this is to write down the same part repeatedly in the program, but it is laboureous. The reason is that it will result in a very large program. This method is also not feasible, because sometimes it is not known how many times a particular portion will be repeated. The best solution for this is to use loops.

11.1 Loops

A loop is a technique which executes an instruction or group of instructions repeatedly. It is also observed that a combination of IF and GOTO statement can also be used as a loop.

The following two types of loops are commonly used:

i) Counter Loop
ii) Controlled Loop

11.1.1 Counter Loop

Counter loop is used to execute a set of statements for a specific number of times. In BASIC, FOR-NEXT statement is used for this purpose.

FOR-NEXT Loop.

This is a loop, which executes the set of statements written between FOR and NEXT statements.

Syntax:

Line no.FOR index=	constant variable expression	TO	constant variable expression	STEP	constant variable expression	

Body Statements

Line no. NEXT Index

Line no. denotes line number and Index denotes index variable. An index variable contains the initial and the final values between which the body of the loop is executed. The initial and final values may each be constant, variable or an expression. The NEXT statement performs two operations. First, it compares the value of the index variable for its final value and secondly, it transfers control back to the start of the loop or to the statement after the loop. Option STEP is used to increase or decrease the interval between the initial and the final value of the index variable. The default interval is one.

Example 1:

```
10 SUM = 0
20 FOR I = 1 TO 10
30 SUM = SUM + I
40 NEXT I
50 PRINT "THE SUM OF 10 NUMBERS IS", SUM
60 END
```

The above program will display the result as:

The som of 10 numbers is 55.

Example 2:

```
10 A = 1
20 B = 9
30 FOR I = A TO B STEP 2
40 PRINT I
50 NEXT I
60 END
```

The above program will display the numbers 1 3 5 7 9 on the screen as the STEP size is two.

If the same program is executed with a slight modification in line number 30 as :

30 FOR I = B TO A STEP - 2

Then the output on the screen will be 9 7 5 3 1.

The following rules should be observed while using a FOR-NEXT loop:

1) The control cannot be transferred from outside of the loop to the inside of the loop.
2) The control can be transferred from inside the loop to outside of the loop.
3) The control cannot be transferred from anywhere to the NEXT statement.

11.1.2 Controlled Loop

Controlled loop is used to extend the statements or the set of statements till a specific condition is satisfied. In this loop, the number of repetitions is not pre-defined. It depends upon the given condition.

In basic, WHILE-WEND statement is used for this purpose.

Note: All versions of BASIC do not support WHILE-WEND structure.

WHILE-WEND Loop

This is a loop similar to the FOR-NEXT loop. The only difference is that this loop is used when the number of iterations is not known in advance to the programmer i.e., it is a Post-test loop.

Syntax:

Line no. WHILE < condition >

Body Statements

Line no. WEND

Example 1:

```
05 REM TO DISPLAY "PAKISTAN ZINDABAD" FOUR TIMES
10 C = (1)
20 WHILE C < 5
30 PRINT "PAKISTAN ZINDABAD"
40 C = C + 1
50 WEND
60 END
```

Example 2:

```
10 REM THIS PROGRAM FINDS THE SUM OF 1,2, ...., 10
20 N = 0
30 SUM = 0
40 WHILE N < 10
50 N = N + 1
60 SUM = SUM + N
70 WEND
80 PRINT "SUM FROM 1 TO 10", SUM
90 END
```

11.2 Nested Loops

It is sometimes necessary to enclose a loop within another loop. Such loops are called nested loops. The inner (nested) loop must begin with its own FOR - TO statement and end with its own NEXT statement. Each inner loop should be completely enclosed with in an outer loop. An outer loop can have several inner loops, which may or may not be nested.

The following example illustrates nested loop structure:

Example :

```
10 FOR  M = 1 TO 5
20 FOR  N = 1 TO 3
30 PRINT M : N
40 NEXT N
50 NEXT M
```

The line 20 to 40 is completely inside the outer loop (line 10 to 50)

SUMMARY

1. A loop is a technique which executes an instruction or group of instructions repeatedly.

2. There are two commonly used types of loops i.e., counter loop and controlled loop.

3. In the counter loop, the number of repetitions of statements is known in advance. FOR-NEXT statements matches a counter loop.

4. The controlled loop helps to extend the set of statements until a specific condition is satisfied. While - Wend is a controlled loop.

EXERCISE

1. What is the purpose of FOR-NEXT loop?
2. Illustrate the use of Nested loop (s) with atleast two examples.
3. What is the main difference between the counter loops and controlled loops?
4. Write a program to find the sum of 1 3 5 7 .... 99.
5. Write a program to compute the product of first ten even integers.
6. Use WHILE-WEND loop to find the sum of 1/4, 1/6, 1/8, ..., 1/100.
7. Fill in the blanks:

   i) Every FOR should have its ..... in a loop.

   ii) The default value of STEP option is .....

   iii) The control ..... be transferred from outside the loop to inside of the loop.

   iv) In controlled loops, WHILE ends with .....

   v) A loop within another loop is called .....

ARRAYS

12.0 Introduction

An array is a sequence of related objects in computer's memory, each of which is identified uniquely by its subscript. Suppose we have an object 'B' having 5 values, then the array structure for this will be :

Location	1	2	3	4	5
B	100	200	300	400	500
Addresses	B(1)	B(2)	B(3)	B(4)	B(5)

Fig. 12.1 (Diagrammatic representation of an Array)

Here B is the array name which is common for all related objects and 1,2,....,5 are the subscripts used to identify the array locations.

So B(1), B(2),....,B(5) are the addresses in the memory where the value will be stored. The contents of these addresses must be of the same type of the defined array i.e., either string or numeric: as B(1) = 100; N$(1) = "ALI".

We can place some values in these locations or memory addresses as :

B(1) = 100, B(2) = 200

This shows that in the array B the first location contains a value 100 and second location 200, and can be referenced by B(1) and B(2).

If we write:

B(3) = B(1) + B(2)

This means that in array B, we have a third location i.e., B(3) and its content will be 300, which is the sum of the contents of location one and two of array B. The primary objective of an array is addressability. That is, we can easily find the addresses of an array by incrementing or decrementing its subscripts.

Arrays are also used to make the programmer's life easy, by defining a single array name instead of defining several names for storing different values.

Example:

```
10 REM TO SEND ELEMENTS AND ARRAY FROM DATA STATEMENT
20 FOR C = 1 TO 10
30 READ X (C)
40 NEXT C
50 REM PRINT THE ARRAY
60 PRINT "ELEMENT", "VALUE"
70 FOR C = 1 TO 10
80 PRINT C, X (C)
90 NEXT C
100 DATA 8, 99, 27, 3, 19, 41, 5, 5, 0, 12
110 END
```

12.1 Subscripted Variables

These are the basic components of an array. Unlike the simple variable, a subscripted variable is a special type of variable used in programming. The general form of a subscripted variable contains the variable name followed by a subscript enclosed in parentheses i.e.,

Syntax:

Variable (Subscript)

Example:

P(1), P(2), etc.

A subscript can be a zero, any positive integer or expression with integeral value.

Following are examples of valid subscripted variables:

P(1,2) P(A) P(A + B) P(A * B * C) P(A/B)

A subscript with negative sign and non-integral values are not allowed. However, a subscript having non-integral value is truncated by some computers such as P(9.8) is treated as P(9).

If an expression is used as a subscript which yields larger positive integeral value than the specified size, an error will occur. A subscripted variable is used in a program like a simple variable. The subscripted variables may be of two types: integer and string. For example, P(1) is an integer subscripted variable and can only be assigned an integeral value. N$(1) is a string subscripted variable and can only be assigned the string value like :

N$(1) = "PAKISTAN"

12.2 Dimension Statement

Dimension statement designed by DIM, tells the computer how many items will be stored in a particular array in the memory. This statement is needed, whenever we try to store more than 10 values in a single array in the memory. For this purpose, we identify the total number of locations to be reserved in the computer's memory.

Syntax:

Line no. DIM NAME OF ARRAY (size), variable (size),.....

Line no. DIM variable(size), (size).....

The dimension statement declares the array length. More than one arrays can be declared in a single statement with the array name separated by commas, and a two dimensional array must have two subscripts separated by comma. A subscript can be an integer or a variable or an expression having an integeral value.

Example 1:

```
10 DIM X (90)
20 DIM A(20), B(30), C(40)
30 DIM D(22)
40 DIM E(10, 10)
```

In this example 90 locations are reserved for array X, 20 for A, 30 for B, 40 for C, 22 for D and 100 for the two dimensional array E in the memory. Less than 10 items need not to be dimensioned in a program, because BASIC automatically reserves 10 locations for one dimensional array and 100 for a two dimensional array. However, it is good programming practice to explicitly

declare all arrays. A dimension statement can be placed any where in a program but before the array manipulation.

Example 2:

```
10 DIM A(15)
20 FOR J=1 TO 15
30 READ A (J)
40 NEXT J
50 LAR = A (1)
60 FOR K = 2 TO 15
70 IF A (K) > LAR THEN LAR = A (K)
80 NEXT K
90 PRINT "THE LARGEST NUMBER IS = ", LAR
100 DATA 50, 40, 13, 10, 60, 45, 7, 100, 9, 15, 80, 90, 1,2,7
110 END
RUN
THE LARGEST NUMBER IS = 100
```

Example 3:

```
10 REM TO READ RNO AND STUDENT SAJID IN ARRAYS
20 DIM RNO (12), N$(12)
30 FOR I = 1 TO 12
40 INPUT "ENTER RNO", RNO (1)
50 INPUT "ENTER STUDENT NAME", N$(I)
60 NEXT I
70 REM ** PRINTING THE ARRAYS **
80 PRINT "RNO" , "STUDENT - NAME"
90 FOR I = 1 TO 12
100 PRINT RNO (I), N$ (I)
110 NEXT I
120 END
```

SUMMARY

1. An array is a group of related memory locations.
2. Arrays are used in many applications.
3. The elements (item) of an array can be either string or an integer quantity.
4. Array names are selected for subscripted variables which contain the variable name followed by a subscript value enclosed with in parentheses.
5. DIM is used in BASIC for dimension statement.
6. DIM statement reserves memory locations for a particular array.
7. The subscripted variable name used in DIM statement indicates the type of the data and its length corresponds to the number of items within the parentheses.

EXERCISE

1. Why do we use an array in programming? Give its advantages over other methods.
2. What is a subscripted variable? Write all essential rules for it.
3. What is the purpose of DIM statement?
4. State which of the following are true or false:
   i) One dimensional array is also called a list.
   ii) DIM statement may be placed anywhere in a program.
   iii) Negative sign is allowed in subscript.
   iv) P4 is a subscripted variable.
5. Write a program to READ ten numbers in an array and print their average.
6. Write a BASIC program to send any fifteen values in an array. Find out their sum and print it.
7. Read the names of ten countries and their capitals in arrays and display them on the screen.

SUBROUTINES AND FUNCTIONS

13.0 Introduction

Sometimes, it is desirable to use a set of statements repeatedly in various parts of a program. So we have to repeate the same statements more than once. This affects the efficiency and economy of the program. Subroutines overcomes this problem.

13.1 Subroutines

A subroutine itself is a small-program consisting of set of statements which are usually written within a main program. The main purpose of subroutines is to avoid duplication and modularize (break into small logical parts) a program. A subprogram is written once, and can be CALLed anywhere in a program. A subprogram should have the facility of returning control back to the next executable statement just after the subprogram calling statement.

There are two main types of subprograms: The function subprograms and the subroutine subprograms. A function is used to calculate and return a single value whereas, a subroutine can be used to calculate and return many values. These are discussed below:

13.1.1 Function Subprograms

The BASIC language contains function subprograms which perform arithmetic and string operations which make programming much simpler in many situations and reduce efforts involved in programming.

There are two types of function subprograms:

1) System-defined functions
2) User's defined functions

1) System-defined Functions

There are certain arithmetic and string operations that most computers can perform without requiring the programmer to write a program for them. Basically, these are programs which have been written by someone else and have been incorporated in the system. These functions are also called standard functions, library functions or built-in functions. These functions are identified by three-letter names. Some commonly used functions are discussed below:

i) ABS(X) Function

The ABS(X) function provides the absolute value of an input parameter. If X is greater than or equal to 0, ABS(X) will be equal to X. Alternatively, if X is less than 0, ABS(X) will be equal, −1 times X. It means that ABS(X) will result in the absolute value of X irrespective of the sign of X.

Example:

X	ABS(X)
−5.9	5.9
15.6	15.6
−2.0	2.0
1.3	1.3
−0.6	0.6

ii) INT(X) Function

INT(X) is called the integer function. This function will print the greatest integer that is less than or equal to its argument.

Example:

X	INT(X)
8	8
6.1	6
8.9	8
0.4	0
−2.8	−3
−2.3	−3
1.3	1
−1.3	2

It should be noted that the value of INT(8.9) equals 8 and not 9. The reason is that the function does not round off; it always truncates.

iii) SQR(X) Function

The SQR(X) function gives the square root of a number. The value of the number inside the parentheses must be positive.

Example:

X	SQR(X)
2	1.414214
3	1.7321051
4	2
5	2.236068
6	2.44949
7	2.645751
8	2.82827
9	3

iv) SIN(X) Function

The SIN(X) function is used to calculate the trigonometric sine of an angle X, in radians. To convert the angle X in radian, use (X*π/180).

For example, SIN (90*(3.14159/180)) = 1.

v) COS(X) Function

The COS(X) function is used to calculate the trigonometric cosine of an angle X. X must be in radians.

For example, COS(0*(3.14159/180)) = 1

vi) TAN(X) Function

The TAN(X) function is used to calculate the tangent of an angle X. The value of angle X must be in radians.

For example, TAN(45*(3.14159/180)) = 1

vii) TAB Function

As we have studied about the Comma operator in the PRINT statement which is used to print at certain column positions on the screen. These printing positions are fixed on the screen, but if our requirements

demand some printing positions which are not accessible by comma, then we will use the TAB function.

Tab is used to print at a specific column position in the first available line after the instruction on the screen.

Example:

```
PRINT TAB (5); "NAME" TAB (15); "TELEPHONE"
Col    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 ...
              ↓                          ↓
              NAME                       TELEPHONE
```

viii) SPC Function

SPC is used to skip certain number of spaces before a character in a PRINT statement.

Example:

```
PRINT SPC (5); "NAME" SPC (15); "TELEPHONE"
Col 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 ...
            ↓                                                      ↓
            NAME                                                   TELEPHONE
```

ix) LEN Function

LEN (X$) returns the number of characters that are contained within the string X$.

Example:

```
10 LET X$ = "ABCDEFHIJK"
20 PRINT LEN (X$)
```

Since there are ten characters in the string X$, the numeric constant 10 will be printed.

x) VAL Function

VAL (X$) returns a numeric value of a string containing numbers. This will convert a string into an equivalent number if the string X$ contains only digits.

Example 1:

```
10 LET X$ = "12345"
20 LET Y = VAL (X$)
30 PRINT Y
```

This program will print the numeric constant "12345". The value of Y, extracted from a string can also be used in an arithmetic expression.

Example 2:

```
10 LET X$ = "12"
20 LET Y$ = "12"
30 LET A = VAL(X$)
40 LET B = VAL(Y$)
50 LET C = A + B
60 PRINT C
```

This program will add the values of A and B and store the result in C, and the value of C will subsequently be printed after the execution of the program. In this case, the resultant value is 24.

xi) LEFT Function

LEFT$(X$,Y) returns the left most Y characters from the string X$. In this case, Y should be an integer.

Example 1:

```
10 LET X$ = "ABCDE"
20 PRINT LEFT$(X$,3)
```

After the execution of the program, it will output the first three characters of the string X$ i.e., ABC

Example 2:

```
10 LET Y = 3
20 PRINT LEFT$ ("ABCDEFG",Y)
```

The output will be first three characters i.e., ABC.

xii) RIGHT Function

RIGHT$ (X$,Y) returns the right most Y characters from the string X$. In this case, Y should be an integer.

Example 1:

```
10 LET X$ = "PQRSTUVWXYZ"
20 PRINT RIGHT$ (X$,5)
```

Since the last five characters of X$ are VWXYZ, therefore these will be displayed on the screen.

Example 2:

```
10 LET Y = 3
20 PRINT RIGHT$ ("ABCDEF",Y)
```

The output will be the right-most three characters i.e., DEF.

xiii) CHR Function

CHR$(Y) will return the character for the given ASCII code Y. Y is a numeric constant, a variable or an expression that has a value between the limits of the ASCII codes i.e., 0 < = Y < = 127. The function CHR serves the opposite to ASC and is useful in getting output ASCII control characters as a part of the PRINT STATEMENT.

Example 1:

```
10 PRINT CHR$(65)
```

This will result in the character A being printed since the ASCII code for A is 65.

Example 2:

```
10 PRINT CHR$(7)
```

This will result in the warning bell since the ASCII code for activating the bell is 7.

xiv) ASC Function

ASC (X) returns the ASCII (American Standard Code for Information Interchange) code of the character within the argument or, in case of a string, the ASCII code of the first character of a string.

Example 1:

```
10 LET X$ = "A"
20 PRINT ASC (X$)
```

The ASCII code for character "A" is 65. Therefore when the above program is executed, the value 65 will be printed.

Example 2:

```
10 Print ASC ("CAR")
```

The ASCII code for the first character C is printed, this will result in the value 67 being printed. Similarly the other character can be printed with the help of ASCII code table (13.1)

xv) DEF FN Statement

This statement is used to define and name a function subprogram written by the user.

Syntax:

DEF FN name [argument] expression

In the above syntax the *name* must be a legal variable name. This becomes the name of the function. The *arguments* consist of those variable names in the function definition which are to be replaced when the function is CALLed. The items in the list are separated by commas. The expression is a demonstration that performs the operation of the function. It is limited to one statement.

In the DEF FN statement, arguments are used only to define the function; they do not affect program variables that have the same name. A variable name used in a function definition may or may not appear in the argument. If it does, the value of the parameter is supplied when the function is CALLed. Otherwise, the current value of the variable is used.

A function can be used in the expressions of a program like a variable. The exact name of the function must be given along with the appropriate arguments.

The variables in the argument represent, on a one-to-one basis, the argument variables or values which are to be given in the function call.

User-defined functions may be numeric or strings. If a type is specified in the function name, the value of the expression is forced to that type before it is returned to the calling statement. If a type is specified in the function name and the argument type does not match, an error message is given.

A DEF FN statement must be executed before the function defines may be CALLed. If a function is CALLed before it has been defined, an error message is displayed.

Example:

```
10 RAD = 1
20 SUR = 1
30 DEF FNFSUR (X, Y) = X ^ 2 / (Y *3)
40 PRT = FSUR (RAD, SUR)
50 PRINT RAD, SUR, PRT
60 END
```

Table 13.1 (The ASCH Character Set)

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
032	blank	064	@	096	?
033	!	065	A	097	a
034	"	066	B	098	b
035	#	067	C	099	c
036	$	068	D	100	d
037	%	069	E	101	e
038	&	070	F	102	f
039	'	071	G	103	g
040	(	072	H	104	h
041	)	073	I	105	i
042	*	074	J	106	j
043	+	075	K	107	k
044	,	076	L	108	l
045	-	077	M	109	m
046	.	078	N	110	n
047	/	079	O	111	o
048	0	080	P	112	p
049	1	081	Q	113	q
050	2	082	R	114	r
051	3	083	S	115	s
052	4	084	T	116	t
053	5	085	U	117	u
054	6	086	V	118	v
055	7	087	W	119	w
056	8	088	X	120	x
057	9	089	Y	121	y
058	:	090	Z	122	z
059	;	091	[	123	{
060	<	092	\	124	|
061	=	093	]	125	}
062	>	094	^	126	~
063	?	095	_	127	⌂
L					

Note: ASCII assigns numeric values from 0 to 225 to each character that may be represented on the computer. The first 32 characters and the last character are control characters.

13.2 The GOSUB and RETURN Statement

A GOSUB is used to invoke a subroutine.

Syntax:

Line no. GOSUB line no.

The GOSUB statement executes the subroutine starting from the given line number.

Example:

```
10 GOSUB 100
20 PRINT "LAHORE"
30 PRINT "PAKISTAN"
40 END
100 PRINT "TECHNICAL WING"
110 PRINT "PUNJAB TEXT BOOK BOARD"
120 RETURN
```

The above program will transfer control to the subroutine at line number 100, when the line 10 is executed. In the subroutine, line 100, 110 and 120 will be executed. On the execution of line 120 (the RETURN statement), control is transferred back to line 20. This is the line just after the statement from where the subroutine was CALLed.

A RETURN statement shows a formal end of the subroutine and transfers control back to the line following the GOSUB statement. The result of the above program will be as follows:

```
TECHNICAL WING
PUNJAB TEXT BOOK BOARD
LAHORE
PAKISTAN
```

Note the difference between a GOSUB and a GOTO statement. A GOTO statement transfers control to another point without keeping track of the next sequentially executable statement and therefore, in general, the control never transfer back to the following statement. In case of GOSUB statement the control is always returned back to the next statement following the GOSUB statement when execution of the subroutine is finished.

Example:

```
10 REM TO CALCULATE THE SQUARE AND SQUARE ROOT OF X
20 PRINT "X", "X ^ 2 ", "SQR (X)"
30 LET X = 4
40 GOSUB 120
50 LET X = 9
60 GOSUB 120
70 LET X = 16
80 GOSUB 120
90 LET X = 25
100 GOSUB 120
110 END
120 REM START SUBROUTINE
130 PRINT X, X^2, SUR(X)
140 RETURN
```

Line number 20 prints the heading for the table. Lines 30, 40, 50, 60, 70, 80, 90 and 100 assign a value to X and send the computer to the subroutine starting in line 120. Line 130 print the results, and line 140 sends the computer back to the main program.

X	X^2	SQR(X)
4	16	2
9	81	3
16	256	4
25	625	5

13.3 Advantages of Writing Subprograms

The advantages of writing a subprogram are given below:

i) Some subprogram can be CALLed anywhere in the program without rewriting it again and again.

ii) Debugging (removing errors) of a program containing subprograms is faster than a program doing the same job without subprograms.

iii) Documentation of a program having subprograms is simpler than the program without subprograms.

iv) Proper use of subprograms break up a big program into small logical parts (modularization) which makes understanding and modification of the program quite easy.

SUMMARY

1. A function is used to calculate and return a single value.
2. There are two types of fuctions: Library functions and User's defined functions.
3. Library functions are pre-defined operations provided in BASIC.
4. User-defined functions are subprograms written by the user(s).
5. Commonly used library functions are ABS, INT, SQR, TAB, SPC, CHR and ASC function.
6. DEF FN statement can be used for numerals as well as strings.
7. A subroutine is used to avoid repetition of small programs within a main program.
8. A SUBROUTINE SUB-PROGRAM can be used to calculate and return more than one values.
9. GOSUB, ON-GOSUB and RETURN statements are used in developing subroutines.

EXERCISE

1. Distinguish between function and subroutine.
2. What is the purpose of DEF statement?
3. How is DEF statement different from GOTO statement?
4. What is the purpose of system defined function or library function?
5. Explain atleast five library functions?
6. What would be displayed by each of the following PRINT statement:
   a. PRINT SQR (36)
   b. PRINT ABS (-17.99)
   c. PRINT INT (5.6)
   d. PRINT LEFT$ ("RIDDA",4)
   e. PRINT RIGHT$ ("JAVAIRIA",3)
   f. PRINT MIO$ ("MAHAM", 4,3)
   g. PRINT LEN ("FATIMA KULSOOM")
   h. PRINT ASC ("JAWAD")
7. Write a subroutine to find out the functional of any number.
8. Write a subroutine for the table of any number.

GRAPHICS

14.0 Introduction

The use of graphics is to sketch and draw diagrams in different modes either in black and white or in color. This facility is available almost in all languages, depending on the hardware used. In this chapter, we shall discuss different statements for sketching and drawing different types of diagrams, pictures and models.

14.1 Graphics

Graphics is an art to design and produce pictorial representation of information. This facility is also provided in almost all the versions of BASIC language. Graphics depends on the hardware such as input, output and graphic cards (e.g. Color Graphic Adaptor, Video Graphic Array).

The demand for good graphics is increasing day by day. The sharpness of a diagram or picture on the monitor screen depends upon the quality of an image. This quality of the image is known as its resolution. There are two main types of resolutions: high-resolution and low-resolution.

In case of high-resolution, picture or diagram will be more clearer. In other words, we can say that there will be more sharpness due to the closeness of pixels. In case of low-resolution, there will be less sharpness. Monitor depends on a card or adaptor. Generally, there are two types of monitors: monochrome monitor and color monitor.

14.2 Modes

The quality of the Color Graphics Adapter (CGA) Monitor is to provide three different types of modes for displaying picture or diagram on a screen.

The three types of modes are:

i) Text mode ii) Medium - resolution graphics mode
iii) High - resolution graphics mode

14.2.1 Text Mode

In this mode screen has the capacity of displaying 25 lines and each line with 80 characters. This makes a 25×80 matrix. A character or diagram on the screen is made up of tiny dots called pixels. Text mode allows the display of letters, numbers, and all the characters in the ACSII Character Set, but is not used for displaying graphics.

The screen statement SCREEN 0 displays data in the text mode.
The width of the screen can be changed by using the following statement:
Line # WIDTH n
Where n can either be 0 to 40 characters indicating the number of columns.

14.2.2 Medium Resolution Graphics Mode

In this mode, screen statement

SCREEN 1

divides the screen into 200×320 pixels. Each pixel is denoted by (x,y) where x shows the position of the row number and y is the column number. The picture/diagram can be displayed in 4 colors. The colors are numbered as 0,1,2 and 3.

14.2.3 High-Resolution Graphic Mode

In this mode, screen is divided into 200×640 pixels i.e., 640 columns and 200 rows of pixels, which permits more precise drawings.

Notice that the screen coordination is enclosed within parentheses and that the row and column values are separated with a comma. Both row and column numbering begins with zero. Each specific screen coordinate is known as a pixel.

A pointer known as the LPR (Last Point Referenced) denotes the pixel position last referenced by a BASIC statement in the graphics mode. In the text mode, the location of the cursor is stored with a different pointer.

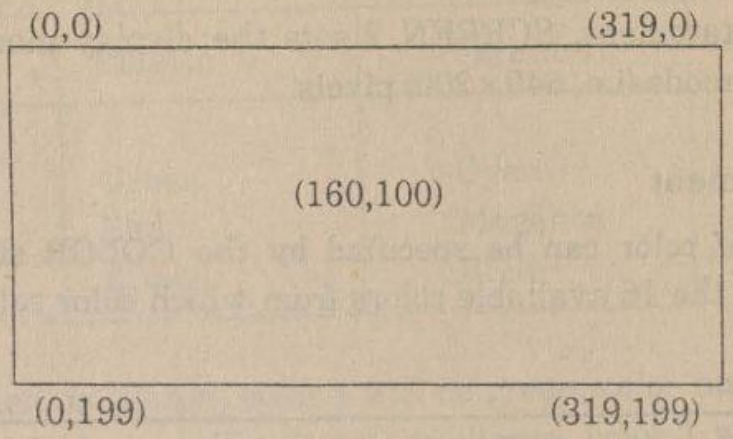


Fig. 14.1 (Medium Resolution Graphics Coordinations)

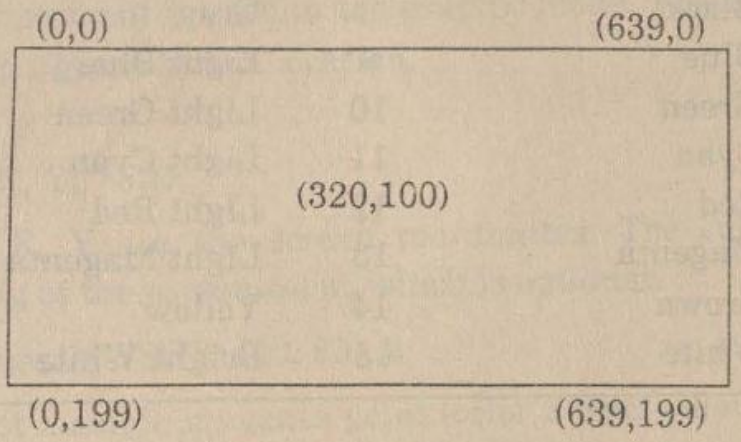


Fig.14.2 (High Resolution Graphics Coordinations)

14.3 Screen Statement

The screen statement allows the programmer to select different display modes.

The simplest form of this statement is line # SCREEN mode.

Where mode indicates the display mode.

Example 1: 10 SCREEN 0

The above statement, SCREEN 0 sets the display mode into the Text mode.

Example 2: 10 SCREEN 1

The above statement, SCREEN 1 sets the display mode to the medium resolution graphics mode (i.e., 320×200) pixels.

Example 3: 10 SCREEN 2

The above statement, SCREEN 2 sets the display mode to the high resolution graphics mode (i.e. 640×200) pixels.

14.4 Color Statement

The choice of color can be specified by the COLOR statement. The following table lists the 16 available colors from which color selection may be made.

Table 14.1

CODE	COLOR	CODE	COLOR
0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Yellow
7	White	15	Bright White

The general form of the COLOR statement in text mode is:

COLOR Foreground, Background, Border

Example:

COLOR 4, 3, 2

The above statement, COLOR 4,3,2 sets the color of the text to red, background to cyan and the screen's border to green.

The COLOR statement cannot be used with high-resolution graphics mode, but can be used in medium resolution graphics mode with the following general form.

COLOR Background, Palette

The Background refers to the background color and must be a number or numeric expression from 0 to 15 and palette is the selection of available palettes and must be either 0 or 1. The palettes are as follows:

Color	Palette 0	Palette 1
1	Green	Cyan
2	Red	Magenta
3	Brown	Blue

If palette 0 is chosen, color 1 will be green, color number 2 will be red, and color 3 will be brown. If palette 1 is chosen, color 1 will be cyan, color 2 will be magenta and color 3 will be blue.

14.5 PSET Statement

PSET statement is used in the graphic mode. This statement is used to draw a point at a given screen location.

Syntax:

PSET (X, Y), color

Where X, Y are the screen coordinates. The color depends on the previous setting of the screen color, which is optional.

For example, PSET (400, 80), 2

Will plot a single magenta point (color 2 from palette 1) at row number 400 and column number 80.

14.6 Line Statement

The LINE statement can be used to draw lines or boxes on the screen. It has the following general form:

LINE (X1, Y1) – (X2, Y2), color, BF

X1, Y1 are the co-ordinates of starting point of the line.
X2, Y2 are the co-ordinates of ending point of the line.
B is for a Box.
F is used for filled box with a COLOR.

The following statement would draw a line on the medium-resolution screen from the point (50, 100) to the point (250, 100).

LINE (50, 100) – (250, 100)

Similarly, the following statement would draw a box and fill it with color 1.

LINE (100, 30) – (250, 150), 1, BF

14.7 Circle Statement

The CIRCLE statement allows the user to draw a circle, an ellipse or an arc on the screen. It has the following form.

CIRCLE (X, Y), Radius, Color, Start angle, End angle, Aspect

Where X and Y are the x - and y - coordinates of the centre of the circle respectively.

Example 1:

CIRCLE (160, 100), 50

The circle is drawn with its centre at coordinates (160, 100) with a radius of 50 pixels.

The following example makes a semi-circle with a default color.

Example 2:

CIRCLE (160. 100), 50, 0, 3.14157

The above statement makes a semi-circle with a default color.

The CIRCLE statement can be used to draw an ellipse rather than a circle.

Example 3:

CIRCLE (160, 100), 50,,,,2

The above statement draws an ellipse with standing up position, as aspect is greater than one.

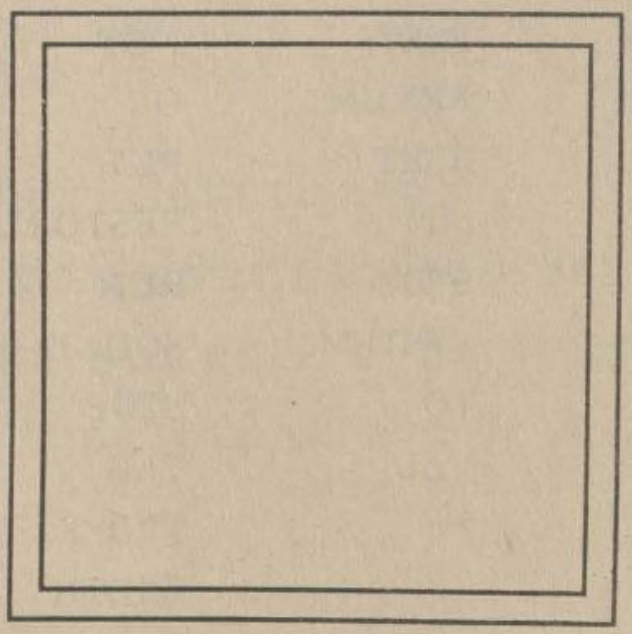
SUMMARY

1. Graphics is an art to design different types of pictures and diagrams.
2. The quality of image is called resolution.
3. There are three modes, text, medium-resolution, graphics and high-resolution graphics mode.
4. In text mode, characters are displayed in terms of 80 or 40 columns and 25 rows.

5. In medium-resolution graphics mode, the screen is divided into a matrix consisting of 320×200 points.
6. In high-resolution graphics mode, the screen is divided into 640×200 points.
7. The screen statement is used to select an appropriate screen mode.
8. The choice of color can be specified by the COLOR statement.

EXERCISE

1. Define graphics. Distinguish the I/O devices commonly used for creating pictures and diagrams.
2. Discuss different types of graphic modes.
3. What is a pixel? Can pixels be controlled?
4. Distinguish between medium-resolution graphics and high-resolution graphics.
5. Write a program to draw the circle of radius 150 pixels inscribed in a square.
6. Write a program to draw the square of side 150 pixels inscribed in a circle.
7. Write a program to draw the following square with double lines.



BASIC RESERVED WORDS

(USED IN THIS BOOK)

APPENDIX - A

ABC	AND	ASC	ATN	AUTO
CIRCLE	CLOSE	CLS	CHR$	CINT
COS	DEF	DELETE	COLOR	CONT
DATE$	ELSE	END	DIM	DATA
EDIT	EXP	FIELD	EOF	DRAW
END	FOR	GET	FILES	ERASE
FN	IF	INKEY$	GOSUB	FIX
HEX$	KEY	KILL	INPUT	GOTO
INT	LINE	LIST	LEFT$	INPUT
LET	LOG	LPRINT	LLIST	LEN
LOCATE	NEW	NEXT	LSET	LOAD
PRINT	ON	OPEN	MOD	MID$
READ	PRINT#	PSET	NOT	MOTOR
RIGHT$	REM	RENUM	OR	OCT$
SCREEN	RND	RSET	PUT	PLAY
SPC	SGN	SIN	RESTORE	RANDOMISE
STRING$	SQR	STEP	RUN	RETURN
THEN	SWAP	SYSTEM	SOUND	SAVE
USING	TIME$	TO	STOP	SPACE$
WRITE#	USING	VAL	TAB	STR$
	**	**	TROFF	TAN
			WIDTH	TRON
			**	WRITE
				**

The Quaid, Said

"You must devote yourself whole-heartedly to your studies, for that is your first obligation to yourselves, your parents and to the state."

قومی ترانہ

پاک سر زمین شاد باد　　کشورِ حسین شاد باد
تو نشانِ عزمِ عالی شان　　ارضِ پاکستان

مرکزِ یقین شاد باد

پاک سَر زمین کا نظام　　قوتِ اُخوتِ عوام
قوم ملک سلطنت　　پائندہ تابندہ باد

شاد باد منزلِ مُراد

پرچمِ ستارہ و ہلال　　رہبرِ ترقی و کمال
ترجمانِ ماضی شانِ حال　　جانِ استقبال

سایہ خدائے ذوالجلال

8726

Date of Publishing	Edition	Impression	Number of copies	Price
May 2001	1st	11th	10,000	15.15